LARGE SCALE RECONFIGURABLE ANALOG SYSTEM DESIGN ENABLED THROUGH FLOATING-GATE TRANSISTORS

A Dissertation
Presented to
The Academic Faculty

By

Jordan D. Gray

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
July 2009

LARGE SCALE RECONFIGURABLE ANALOG SYSTEM DESIGN ENABLED THROUGH FLOATING-GATE TRANSISTORS

Approved by:

Dr. Paul E. Hasler, Committee Chair
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. F. Levent Degertekin
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. David V. Anderson
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. William D. Hunt
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. Farrokh Ayazi
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Date Approved: April 2009

ACKNOWLEDGMENTS

I would like to thank all of my lab mates who helped me throughout this process. Paul, thank you for your advisement. Ryan, thank for your mentorship. And Ashley, thank you for your patience, your friendship, and your love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This work is concerned with the implementation and implication of non-volatile charge storage on VLSI system design. To that end, the floating-gate pFET (fg-pFET) is considered in the context of large-scale arrays. The programming of the element in an efficient and predictable way is essential to the implementation of these systems, and is thus explored. The overhead of the control circuitry for the fg-pFET, a key scalability issue, is examined. A light-weight, trend-accurate model is absolutely necessary for VLSI system design and simulation, and is also provided. Finally, several reconfigurable and reprogrammable systems that were built are discussed.

CHAPTER 1

OVERVIEW

The mapping from abstract computations to physical implementations is the pursuit of analog and digital designers alike. Analog implementations are typically more difficult to design than digital implementations because the links between computations and analog implementations are weak analogies; multiplications, additions, and other computations break down over signal magnitudes, bandwidths, and temperature and process variations. In a digital design, the links between computations and implementations represent much stronger analogies.

As a result analog design is much more difficult for VLSI system design than digital. Exacerbating the difficulty of analog design is the absence of a malleable, intrinsic memory element. As a result, analog designs dependent on fixed values of computation tend to utilize high-gain feedback loops and matched components, which come at the cost of power and area, respectively. In addition, it is common to use digital-to-analog converters (DACs) to provide well-defined values for computation, a significant area burden.

Non-volatile charge storage, as applied to analog CMOS design, is the key to strengthening the analogy between computation and analog implementation. Beyond simply providing an accurate method for open-loop computation and reducing the area impact to match components through offset removal, the analog memory element provides a means for implementing the biasing for VLSI components that would be unfeasible otherwise.

This work is concerned with the implementation and implication of non-volatile charge storage on VLSI system design. To that end, the floating-gate pFET (fg-pFET) is considered in the context of large-scale arrays. The programming of the element in an efficient and predictable way is essential to the implementation of these systems, and is thus explored. The overhead of the control circuitry for the fg-pFET, a key scalability issue, is examined. A light-weight, trend-accurate model is absolutely necessary for VLSI system design

and simulation, and is also provided. Finally, several reconfigurable and reprogrammable systems that were built are discussed.

In chapter two, I introduce the core analog reprogrammable memory and computational element, the floating-gate transistor. Charge storage is addressed using Fowler Nordheim tunneling for charge removal, while channel hot-electron injection is the primary mechanism used for precise and accurate charge storage. The floating-gate pFET is used throughout this work to enable large-scale reprogrammable and reconfigurable analog systems.

In the third chapter, I discuss how to integrate floating-gate transistors into dense arrays. I focus on isolation issues in addressing a single device in a large array. In previous work, devices were addressed for injection by combining a high-field and the biasing to form a substantial channel. I show that subthreshold conduction is not the dominant parasitic charge movement mechanism over the entire operating range of the device. Further, I suggest how to use switch elements and biasing to eliminate parasitic charge movement.

In the fourth chapter, I provide a more detailed discussion of floating-gate channel hot-electron injection along with a simulator-targeted floating-gate injection model. In order for floating-gate injection to become a ubiquitous analog design methodology, as opposed to a risky technique, a robust model with strong simulation support is necessary. As a result, I extend the model, apply it to Verilog-A, and demonstrate how to use the drain current of a floating-gate transistor to fit the model for simulation.

The fifth chapter represents my effort to explore the use of floating-gate transistors as they apply to the core functionality of a vector-matrix multiplier. Current mirroring using the gate degrades over several orders of magnitude due to device mismatch, which can be addressed by using the source voltage and and a buffer. Continuous-time fixed-current injection is employed to reduce the overhead of modeling and targeting of programmed currents.

I use the sixth chapter to discuss two different targeted, reprogrammable analog systems: a transform imager and an adaptive filter. The imager uses floating-gate transistors

for vector-matrix multipliers and offset removal, while the adaptive filter uses floating-gate transistors in synapses that implement least-mean squared learning.

The floating-gate element is used more broadly in the seventh chapter to implement full-scale reconfigurable analog. The fg-pFET is used for biasing, computation, and connectivity in two different field-programmable analog arrays (FPAAs). In building large-scale reconfigurable analog systems, there is a fundamental choice to be made between the level of reconfigurability and the level of area consumption and associated increases in circuit parasitics. The first system is a generic reconfigurable analog signal processor (RASP). The second system is a reconfigurable analog array of MITEs (RAAM), where a MITE is a multiple-input translinear element. The RAAM trades additional area for an atomic computational element with a clear mapping between algorithms and analog implementation.

Chapter 8 provides a summary of the work completed.

CHAPTER 2

FLOATING-GATE ELEMENTS

First formally conceived in 1967, a floating-gate transistor is named as such because of the electrically isolated material that forms the gate of the transistor. As a methodology, it represents a means for implementing a non-volatile memory element in silicon CMOS technology. The floating-gate transistor is a critical element of modern micro-scale electrical circuitry, as it sits at the core of FLASH memory. And though floating gates are primarily used as a storage mechanism for digital systems, there has been a trend of research and development for floating gates as an analog circuit element over the last 15 years [1, 2, 3, 4, 5, 6, 7]. By understanding the I-V relationship and charge storage issues of the floating-gate transistor, it can be used effectively to enhance analog circuit design and implementation.

2.1 Device characteristics

A floating-gate transistor in its simplest form is a standard MOS transistor with a capacitor in place of a gate contact. The device shown in Figure 1 is an example of a typical floating-gate transistor. Multiple coupling capacitors are often used in designing floating-gate transistors. The current through the device is controlled by the voltages coupling onto the gate node through explicit and parasitic capacitors in conjunction with the charge on the

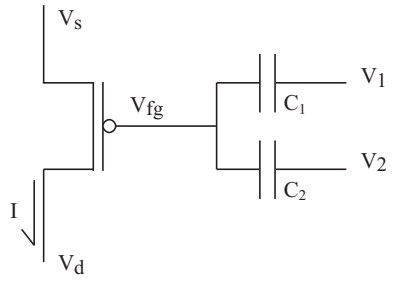


Figure 1. A basic floating-gate schematic with two coupling capacitors.

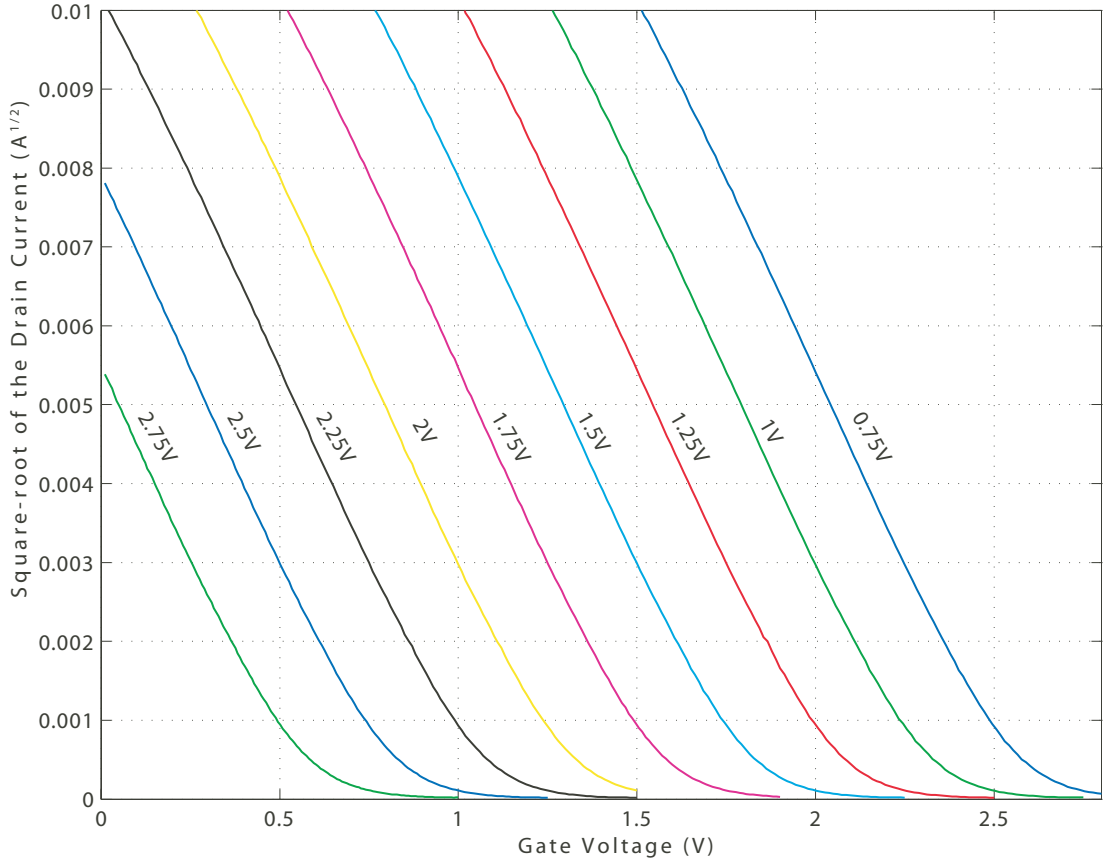


Figure 2. A floating-gate transistor programmed to different threshold values. The effective threshold voltage is given for each curve.

isolated gate region. The relationship between the terminal voltages and drain current of the two-input floating-gate transistor, assuming saturated subthreshold operation, is given by the following equation:

$$I = I_s \, e^{\frac{(V_{DD}-V_s)-\kappa\left(V_{DD}-V_{fg}\right)}{U_T}} e^{\frac{V_{DD}-V_D}{V_A}} \tag{1}$$

where the floating-gate voltage is formulated as the following:

$$V_{fg} = \frac{C_1}{C_T} \cdot V_1 + \frac{C_2}{C_T} \cdot V_2 + \frac{C_d}{C_T} \cdot V_d + \frac{C_s}{C_T} \cdot V_s + ... + \frac{Q}{C_T} \tag{2}$$

or more simply,

$$V_{fg} = \sum_i \frac{C_i}{C_T} \cdot V_i + \frac{Q}{C_T} \tag{3}$$

where Q is the isolated charge and V_i is the i^{th} voltage connected through capacitor C_i to the gate with total capacitance C_T connected to it.

There are at least two important implications of (3): the gate voltage is a function of the charge stored on it, and the gate voltage is a function of any other voltage capacitively coupled to the gate. Because the gate voltage is a function of the charge stored on the floating gate, the I-V curve of the transistor can be shifted to a particular, desirable point. Illustrated in Figure 2 is a series of gate sweeps for a floating-gate device with different amounts of charge stored. The sweeps were performed by using the capacitors C_1 and C_2 in conjunction to couple onto the floating-gate. The result is a single transistor with a wide array of possible effective threshold values, with a threshold given by:

$$V_{th} = V'_{th} + \frac{Q}{C_T} \tag{4}$$

where V'_{th} is the threshold of the same transistor without a floating gate. The implication is that for DC conditions, the current through a floating-gate transistor can be set as precisely as the charge on the floating-gate can be controlled.

The current through a floating-gate transistor under the condition that its terminals are not at fixed potentials is less well defined because of the contribution to the effective gate

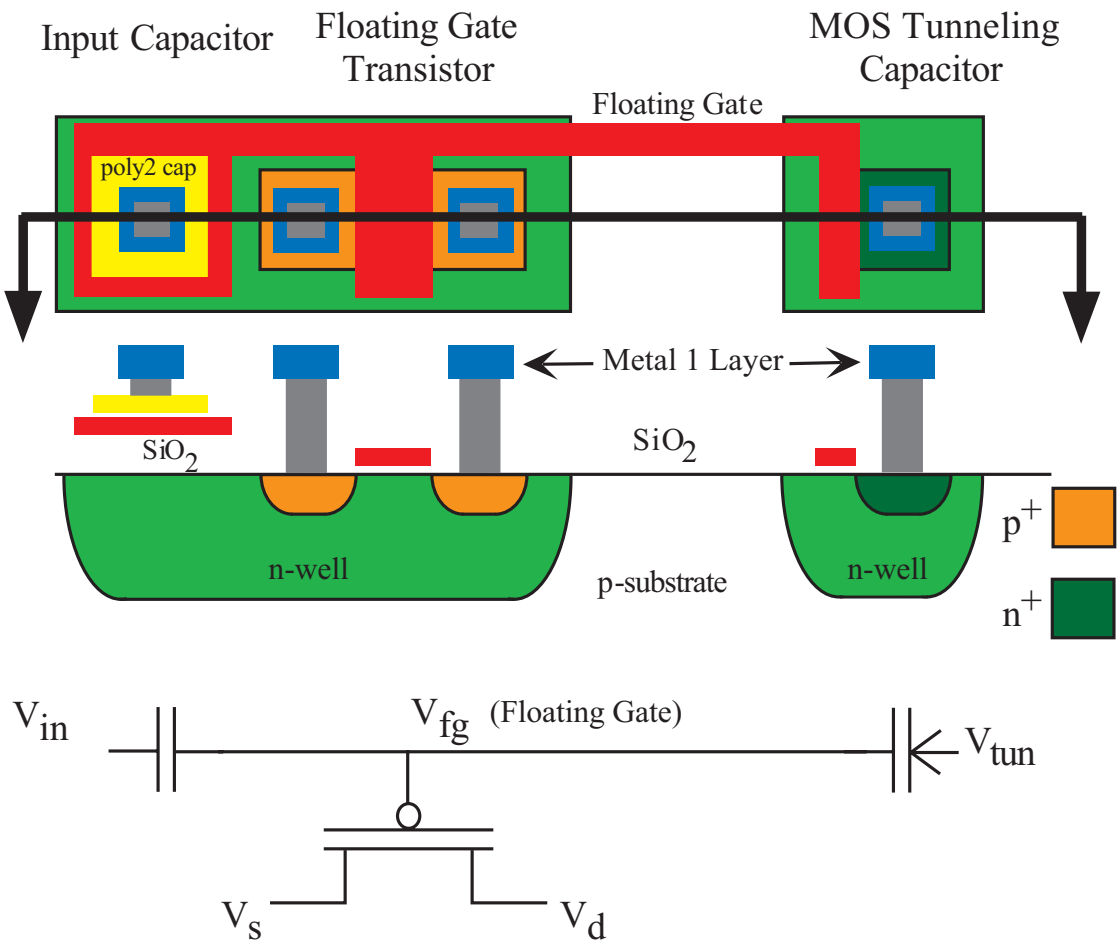


Figure 3. The layout for a floating-gate transistor. The MOS capacitor is used for charge removal, while the drain of the transistor is where charge addition occurs. Both issues are covered in Section 2.2.

voltage from potentials though capacitive coupling. For instance, the Early Effect of a floating-gate transistor is typically dominated by capacitive coupling rather than the length of the transistor. The effect of the drain coupling can be engineered to a nominal factor by decreasing the overlap capacitance or increasing the value of C_T.

The layout for a floating-gate transistor is shown in Figure 3. The MOS capacitor is used for charge removal while the drain of the transistor is where charge addition occurs. Both issues are covered in Section 2.2.

2.2 Floating-gate charge movement techniques

As discussed previously, the I-V characteristic of a floating-gate transistor is strongly defined by the charge stored on the floating gate. In this work, two physical phenomena are purposefully employed to control the value of stored charge: Fowler-Nordheim (FN) tunneling and hot-electron injection.

2.2.1 Tunneling

Electron tunneling is the process by which an electron passes through a barrier rather than traversing the conduction band associated with that barrier. In the case of direct band-to-band tunneling, electrons may pass though a barrier without any assistance. The likelihood of direct band-to-band tunneling is related to the thickness of the barrier. Obviously, oxides that demonstrate appreciable levels of direct band-to-band tunneling are unsuitable for building an electrically isolated gate.

In the case where the oxide thickness is not so thin that spontaneous tunneling dominates, a field-assisted mechanism called Fowler-Nordheim tunneling can be used to modify charge stored on a floating material. Illustrated in Figure 4, electrons on the floating-gate are trapped by the barrier imposed by the SiO_2. By lowering the voltage of the silicon, the energy bands bend in such a way that the electrons see a thinner, triangular barrier. As a result, electrons tunnel through the material. The process is sometimes referred to as a tunneling diode since the current can only move in one direction. As a result, the system is in negative feedback, constantly slowing the rate of tunneling. As current flows through the tunneling diode, the loss of electrons on the polysilicon results in a positive change in potential. The net result is a widening of the triangular barrier and a decrease in tunneling current. In Figure 3, the capacitor between V_{fg} and V_{tun} is the tunneling diode used for purposeful tunneling of charge. In addition to the explicit tunneling diode, a field acting on the gate overlap between the drain and gate can be sufficient to cause FN tunneling. This occurs when the voltage across the gate oxide is raised to the point where electrons pass through the narrowed gate insulator onto the floating gate. Shown in Figure 5, the condition

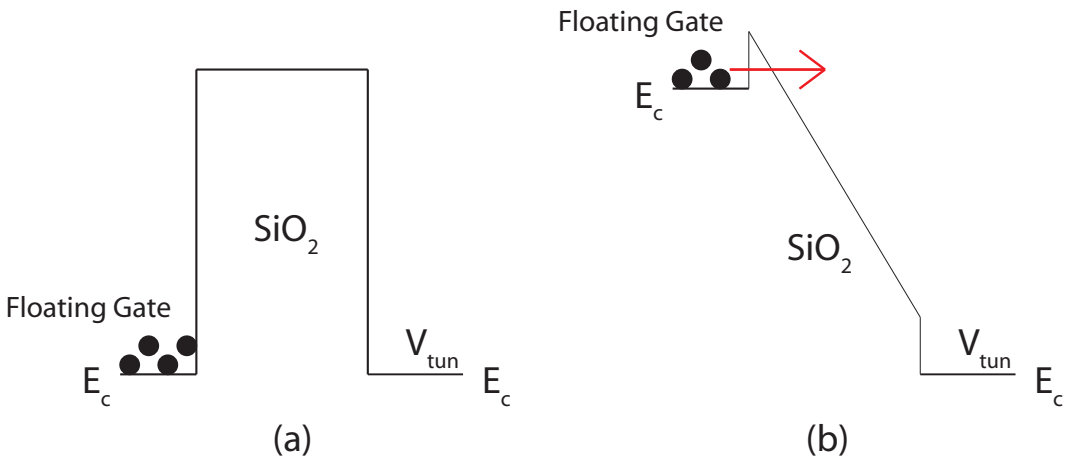


Figure 4. An illustration of Fowler-Nordheim tunneling. (a) Initially, the SiO_2 inhibits tunneling. (b) By creating a significant voltage difference across the barrier, the conduction band bends until carriers can tunnel though the narrow triangular region.

typically arises in a parasitic fashion in arrays of floating-gate transistors when the shared gate is brought to the highest potential possible to minimize subthreshold conduction on columns where the applied drain voltage is near ground.

Practically speaking, controlled tunneling of a floating-gate transistor is accomplished by choosing a particular bias point and then applying tunneling voltage significant enough to implement charge movement. As high voltages tend to be expensive with respect to resource utilization, the bias point can be chosen to decrease the necessarily high tunneling potential. The best case scenario for reducing the tunneling voltage in a circuit is to bring the terminals of the floating-gate transistor to the lowest potential available–ground in a single supply circuit. The change in the floating-gate voltage from the bias point in normal operation to the bias point at ground is proportional to the decrease in the requirement for the maximum tunneling voltage.

2.2.2 Injection

Hot-carrier injection is the process by which a carrier is excited to the point that it can surmount an interface barrier and enter the region of the associated barrier material's conduction band. Common forms of hot-carrier injection include UV light exposure and channel

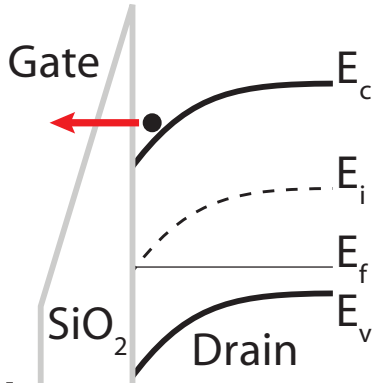


Figure 5. An illustration of Fowler-Nordheim tunneling due to the high field at the gate-drain overlap. The condition typically arises in a parasitic fashion in arrays of floating-gate transistors when the shared gate is brought to the highest potential possible to minimize subthreshold conduction on columns where the applied drain voltage is near ground.

hot-carrier injection [8][9].

In EPROM memory structures, UV light is typically used to erase the device before it is programmed. The dominant charge movement mechanism is injection. UV light generates carriers in the silicon and imparts them with enough energy to promote the carriers into the conduction band of the dielectric that isolates the floating-gate of the EPROM.

Channel hot-carrier injection refers to channel-current injection. A minority carrier traveling from source to drain gains enough energy because of the source-drain field to surmount the gate interface and enter the conduction band of the barrier rather than making it to the drain.

A particular type of channel hot-carrier injection is one where the majority carriers of a transistor are created and subsequently inject due to high-energy minority carriers. In a pFET, electrons resulting from hole impact ionization are provided sufficient energy and trajectory to surmount the barrier presented by the gate oxide. The holes impact ionize in the drain region because of the high field between the channel and drain. The resulting electrons travel out of the drain in the direction of the channel. Those electrons not swept out through the bulk inject into the gate because of the gate to channel field. An illustration of the process is provided in Figure 6a and a band diagram of the process is shown in Figure

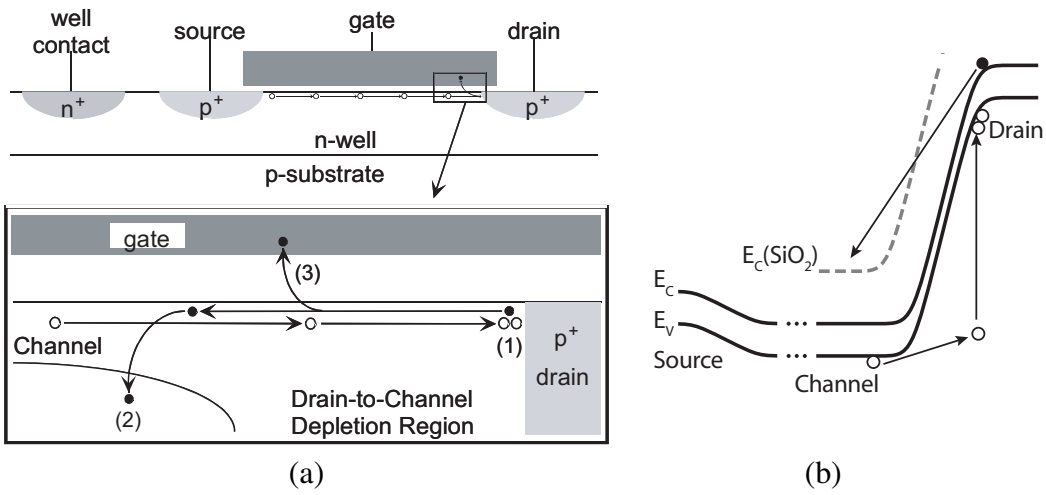


Figure 6. Illustration of channel hot-electron injection in a pFET. (a1) The minority carrier impact ionizes the drain region, creating an electron-hole pair. (a2) Majority carriers are swept out into the bulk. (a3) Because of the gate-to-bulk field, a portion of the high-energy majority carriers inject into the conduction band of the barrier and enter onto the floating-gate. (b) Band diagram of channel hot-electron injection in a pFET

6b. A pMOS transistor is shown since it is the floating-gate device type used in this work. The reasoning behind the use of a pMOS is covered in Section 2.3. It is important to note that while the injection mechanism is well defined by the channel current, subthreshold conduction guarantees that injection will occur so long as the channel to drain potential is large enough. As a result, the subthreshold conduction results in parasitic charge injection.

One other form of parasitic charge injection is a result of high fields that can be present at the isolation junction of a pFET drain. The process, illustrated in Figure 13, occurs when the transistor has a high gate voltage and a low drain voltage and when the area under the gate is accumulated with bulk majority carriers. Any thermally generated electrons that are swept through the high field have the potential to inject into the gate. Exacerbating the situation is the gate overlap of the PN-junction at the inside edge of the drain. The high potential on the gate necessary to reduce subthreshold conduction can deplete or even invert the p+ region under the gate overlap. The severe band bending that occurs leads to band-to-band tunneling of electrons from the valence band to the conduction band. The current, called gate-induced drain-leakage (GIDL) current [10], has the potential to inject

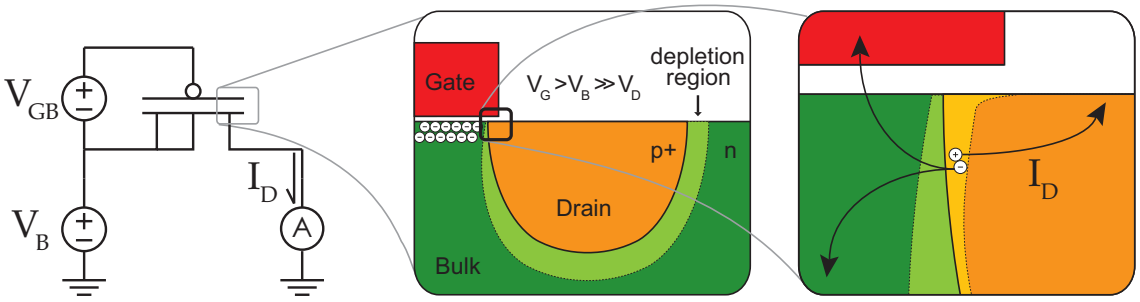


Figure 7. Conditions necessary to generate GIDL current in a pFET. The gate is brought above the bulk, the drain is held fixed at ground. Positive V_{GB} results in accumulation which pinches the depletion region around the drain. The high field and narrow depletion region allow electrons in the valence band of the drain to tunnel though the depletion region, illustrated by light-green and light-orange. When that occurs, holes are generated that move out to the drain. The electrons move toward the gate and bulk in a manner similar to drain avalanche hot carrier (DAHC) injection.

into the floating gate [11].

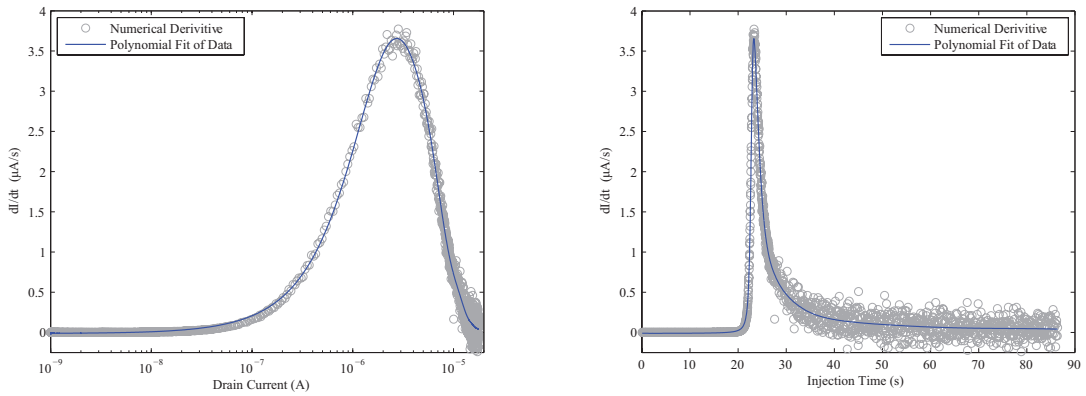
2.3 Programming pMOS transistors

While nMOS transistors are the dominant device choice for FLASH and EEPROM non-volatile digital memory structures, special processing steps are generally required to sustain reliable, consistent operation. On the other hand, pMOS transistors with sufficiently large gate oxides available in standard CMOS processing are well suited for direct floating-gate implementation [12]. As a result, the following programming techniques are expressed for pMOS floating gates on standard CMOS processes exclusively.

The FPAAs discussed in this document rely on a large number of floating-gate transistors. While the programming methods discussed in the following subsections relate only to single transistors, the framework for interacting with arrays of floating-gate transistors is covered in Chapter 3.

2.3.1 Removing electrons

To create the triangular barrier necessary to tunnel electrons off of the floating gate, extremely high voltages must be used. In many cases, the voltage for tunneling exceeds the



(a) (b)

Figure 8. This is experimental data for the time-derivative of current during injection. The V_{SD} was kept small enough that measurement on the order of seconds was possible. (a) Peak injection occurs relative to a particular current level. (b) At a time after peak injection, the rate of current change falls off with approximately a $\frac{1}{X}$ dependence.

breakdown voltage for the active-to-bulk PN junctions in a process, making on-chip instrumentation difficult. In addition, charge movement through Folwer-Nordheim tunneling is less well characterized for floating gates than injection. As a result, it is more convenient to use tunneling as a global erase. Each floating gate has a tunneling capacitor that consists of a MOS-cap, as shown in Figure 3. A MOS capacitor is used for tunneling since it is the highest quality oxide available in a standard CMOS process.

2.3.2 Adding electrons

Channel hot-carrier injection is a common technique for adding electrons to floating-gate nFETs because an electron is the minority carrier to be injected. In a pMOS device, the majority carrier is an electron, so one would expect reasonable current densities resulting from DAHC injection. As a result, the following techniques relate to DAHC injection.

2.3.2.1 Gate-sweep injection

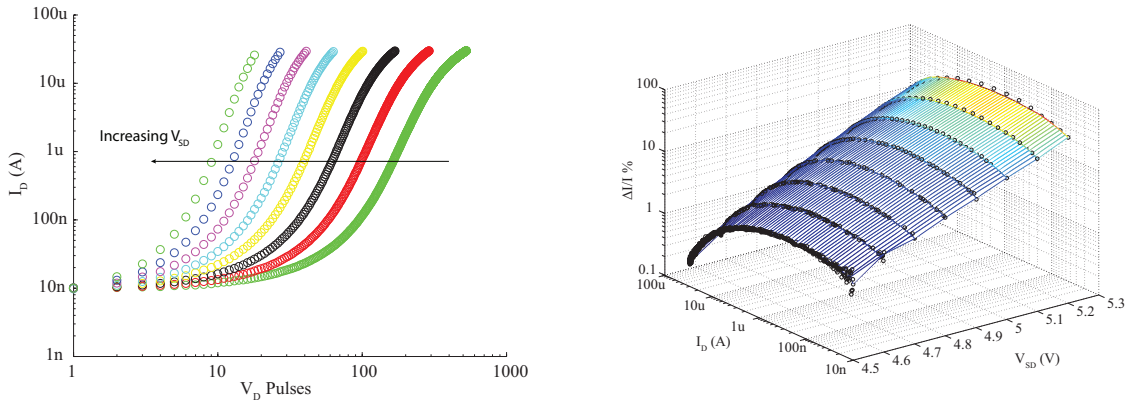
Gate-sweep injection is used when exact current levels are not important and results in high levels of injection. The need for a gate sweep results from the observation of the instantaneous rate of change of the drain current, shown in Figure 8. The instantaneous

change in the current relates to the efficiency of injection. Figure 8a illustrates that peak injection is related to a particular current level that corresponds to a particular effective gate voltage. Moreover, by injecting electrons onto a floating gate, the effective voltage is constantly changing. To counteract the negative feedback from the accumulation of charge, the gate must be constantly moved in order to maintain injection.

Often there is so much charge on the floating gate that it is not possible to inject at maximum efficiency. As a result, a linear gate sweep is not necessarily the best choice. As illustrated in Figure 8b, beyond the maximum injection efficiency, the rate of injection falls off at approximately $\frac{1}{X}$. When injecting a device that cannot be brought back to peak injection current levels, it is necessary to spend a longer time injecting. In particular, it is often beneficial to use a gate sweep with a logarithmic characteristic. The logarithmic curve allows for a higher density of points at higher voltages, counteracting the reduced injection efficiency.

2.3.2.2 Drain-pulse injection

Drain-pulse injection is a characterization-intensive programming methodology and results in very efficient, accurate programming. It works by injecting a floating-gate transistor in short bursts or pulses, and is more completely described in [13]. The illustration in Figure 9a is the first step in the process. A transistor has been injected over a wide range of V_{SD} voltages. Because pulses are used, a derivative is not possible because of parasitic effects relating to the rising and falling edge of the pulse. Instead, the percentage change is used as a means for evaluating the injection efficiency. The data, plotted as black circles in Figure 9b, relates a particular drain current and V_{SD} to a percentage change in the floating-gate transistor current. The data is curve-fit, resulting in the surface of Figure 9b. It represents a mapping from one current level to another. When a particular floating-gate current is necessary, the mapping provides the necessary V_{SD} to reach the new current level.



(a) (b)

Figure 9. (a) The raw data used to characterize drain-pulse injection. (b) The data is replotted as $\frac{\Delta I}{I}$ vs I, the black circles, and curve fit, resulting in the surface shown.

2.3.2.3 GIDL injection

In order to measure the GIDL current, the schematic in the left portion of Figure 7 is used. The bulk and source are tied together, the drain is connected to ground through an ammeter, and the gate is biased above the bulk potential. The resulting drain current for bulk voltages ranging from 6 to 8V and gate-to-bulk voltages from 0 to 3V is shown in Figure 10. The measurement floor of 10 pA is related to the reverse-bias current from the clamp-diodes protecting the drain terminal. Two useful conclusions are apparent from the experimental data. If there is less accumulation around the drain, the GIDL current is reduced. In addition, the bulk-to-drain voltage is also related to the leakage current measured—even with no accumulation, the current increases with bulk voltage.

2.4 Charge retention

The quality of the circuits and systems discussed in this document are proportional to the precision that charge can be injected onto electrically isolated polysilicon. Therefore, a key characteristic of a floating-gate device is its ability to retain charge for long periods of time with minimal leakage. The work in this section has been published and is used to characterize the offset variation of an amplifier over time with respect to charge loss

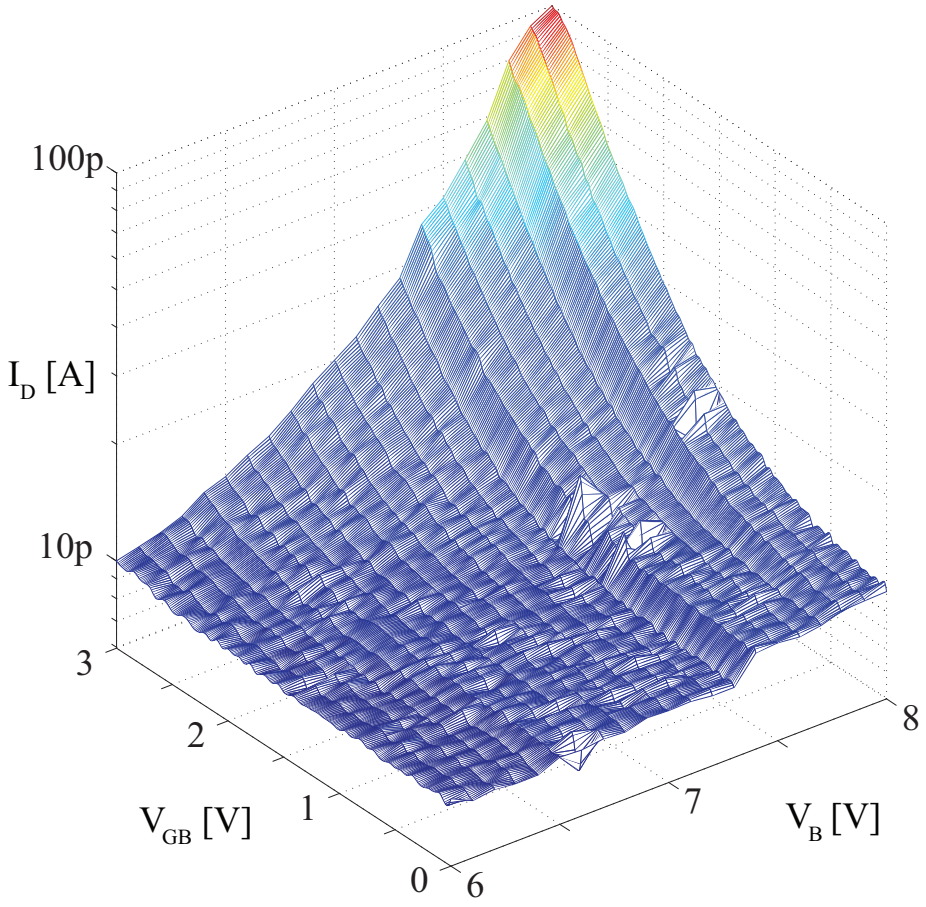


Figure 10. Experimental results for gate induced drain leakage. The bulk and source are tied together, the drain is connected to ground through an ammeter, and the gate is biased above the bulk potential. The resulting drain current for bulk voltages ranging from 6 to 8V and gate-to-bulk voltages from 0 to 3V is shown. The measurement floor of 10 pA is related to the reverse-bias current from the clamp-diodes protecting the drain terminal. Two useful conclusions are apparent from the experimental data. If there is less accumulation around the drain, the GIDL current is reduced. In addition, the bulk-to-drain voltage is also related to the leakage current measured—even with no accumulation, the current increases with bulk voltage.

resulting from a floating-gate offset removal technique [14].

Assuming a high-quality oxide, the mechanism for losing charge over time is thermionic emission [15, 16]. The amount of charge lost is a function of both temperature and time and is given by

$$\frac{Q(t)}{Q(0)} = exp\Big[- t\upsilon \cdot exp\Big(\frac{-\phi_B}{kT}\Big)\Big] \tag{5}$$

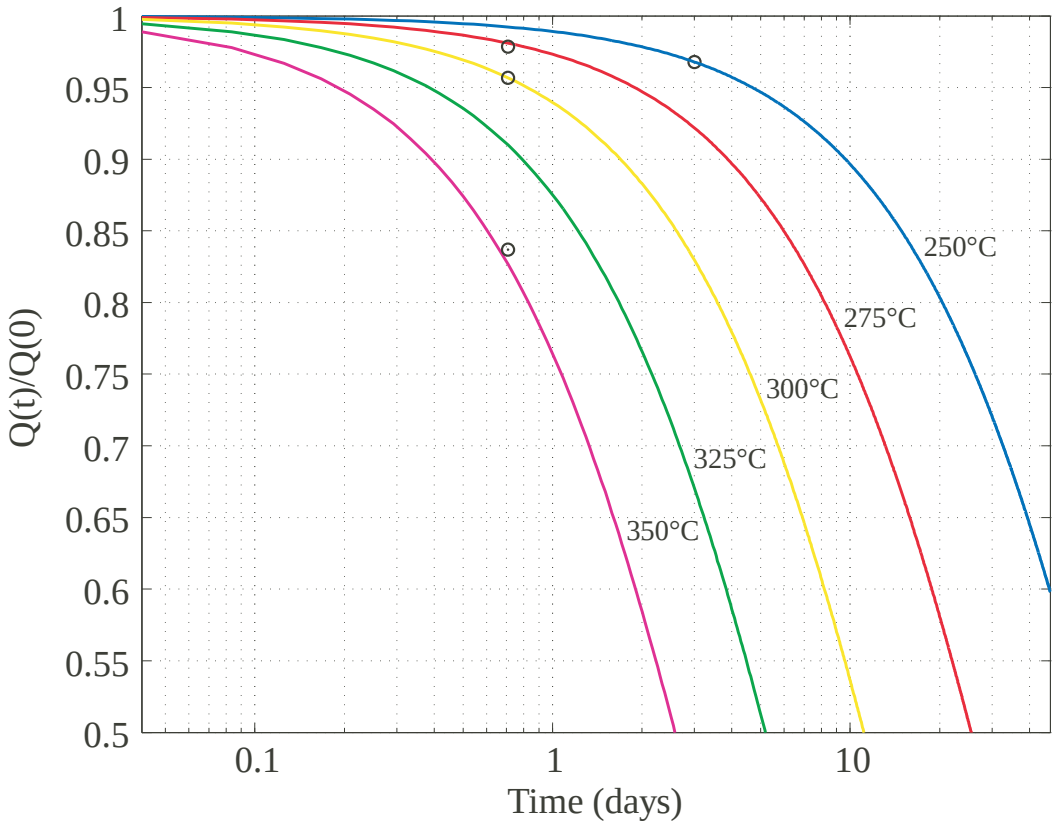
where $Q(0)$ is the initial charge on the floating gate, $Q(t)$ is the floating-gate charge at time t, υ is the relaxation frequency of electrons in poly-silicon, ϕ_B is the $Si - SiO_2$ barrier potential, k is the Boltzmann's constant, and T is the temperature. For normal temperatures, the amount of charge lost is typically very small and difficult to measure. By increasing the temperature, the thermionic emission can be increased to measurable levels.

Knowing the exact charge on a floating gate is not straightforward. An easier way to approach $\frac{Q(t)}{Q(0)}$ is to take advantage of a ratio of threshold voltages before and after programming. By rearranging (4),

$$\frac{Q(t)}{Q(0)} = \frac{V_{th}(t) - V'_{th}}{V_{th}(0) - V'_{th}} \tag{6}$$

The values of parameters υ and ϕ_B were estimated to be $60s^{-1}$ and $0.9eV$ using experimentally measured values of charge loss for different time periods when the devices were exposed to high temperatures ($> 250°C$) for a prolonged period of time.

Figure 11 shows the measured floating-gate charge loss along with a theoretical extrapolated fit using the estimated model parameters. Also in Figure 11 is a summary of the percentage change in floating-gate charge between two floating-gate transistors programmed to different thresholds. Pairs of floating-gate transistors were used to avoid the dependence on measured charge. The two different cases were 10% programming change from initial and 50% programming change from initial. The measured data agrees well with the theoretical prediction, and the trends observed in Figure 11 have been observed across many floating-gate devices. The values in the table inset in Figure 11 have been evaluated using (5) and assuming a subthreshold operation. No significant change can be extrapolated for programmed currents for a period of 10 years at room temperature, indicating good charge



	Programmed 10% change in current			Programmed 50% change in current		
Temperature	**ΔQ/Q**	**ΔV_{fg}**	**ΔI/I**	**ΔQ/Q**	**ΔV_{fg}**	**ΔI/I**
25°C	1e-3%	36.7nV	2e-4%	1e-3%	156nV	9e-4%
90°C	0.62%	16.3µV	0.06%	0.62%	65µV	0.57%
140°C	18.2%	1.8mV	1.8%	18.2%	1.92mV	10.7%

Figure 11. The plot shows the measured charge loss (○'s) plotted with an extrapolated theoretical fit (solid) for different temperatures and time. The table summarizes the percentage change in the floating-gate charge, voltage, and current over ten years for two different cases: (a) 10% **programming change from initial (b)** 50% **programming change from initial.**

retention in floating-gate devices.

CHAPTER 3

FG-PFET ARRAYS

Non-volatile charge storage is a transformational technology in the field of analog integrated circuit design. From enabling the matrix-vector multiplications in an analog pattern classifier and a computational image sensor to facilitating circuit topology transformations and biasing of a field-programmable analog array (FPAA), non-volatile charge storage has been established as a key analog computational system building technique [17][18][19]. In the aforementioned examples, the charge storage is implemented through the use of a floating-gate pFET (fg-pFET) transistor. In the case of the image sensor work, over 3,000 floating-gate transistors are used. The classifier and FPAA contain more than 28,000 and 50,000 floating-gate transistors, respectively. To facilitate the design and implementation of all of those analog memory cells, the floating-gate transistors are arranged in regular arrays, Figure 12, reminiscent of early FLASH memory [20]. The compact floating-gate selection and isolation circuitry employed in such arrays is aggressively sized to minimize area.

3.1 Typical Implementation

A floating-gate array can come in a number of different forms. A good starting point for consideration is the full cross-bar switch matrix. So-called because it looks liked crossed bars, the array topology allows any row to be connected to any column through the weight associated with the programmed charge. Currents sum along a drain line as a result of Kirkov's current law. For a current-mode vector applied along the rows of the array, the drain currents are representative of a matrix-vector multiplication. In the extremes of charge storage, the array is essentially a connectivity grid that allows for reconfigurable topology. Figure 12 is a representation of a 4x7 floating-gate array with the basic programming circuitry included.

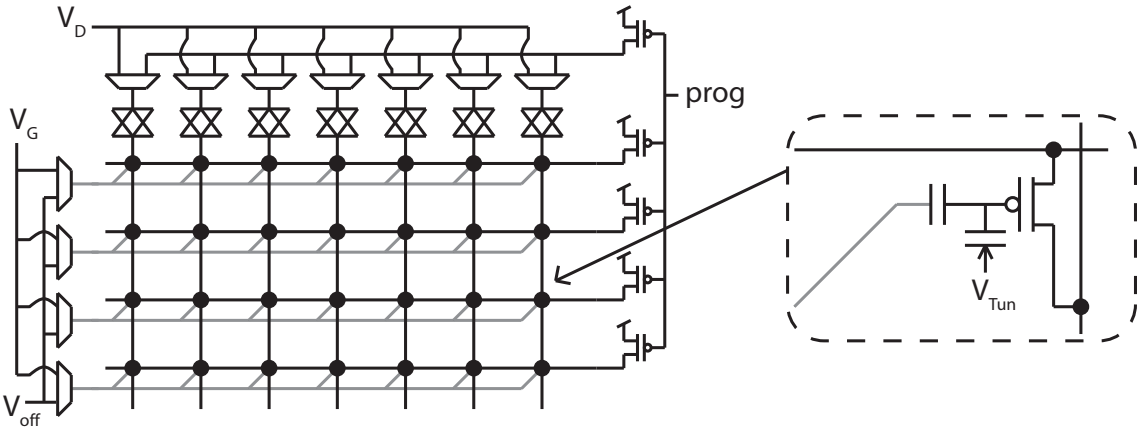


Figure 12. Switch matrix array with basic selection circuitry. Each black dot in the array is a floating gate transistor. The circuitry around the outside of the crossed-bars is necessary for programming the individual floating-gates.

Each black dot in the array is a floating-gate transistor. The circuitry around the outside of the crossed bars is necessary for programming the individual floating gates. When the pass and t-gate controls are de-asserted, rows and columns of the floating-gate transistors are used as switches and the array is said to be in *run* mode. In the case where voltages are asserted along the rows and columns of the matrix, the array is said to be in *prog* mode because variations in the asserted voltages can result in programming of the floating gates. For the purpose of visibility, the logic for setting t-gate and MUX bits is not shown. In addition, the pull-up transistors controlled by the program signal are drawn on the right side instead of the left for visibility as well. Programming circuitry is generally centralized in order to reduce the total area impact.

3.2 Isolation

The goal of array programming is to provide a methodology for controlling the charge on a large number of floating-gate transistors in a spatially and temporally efficient manner. The circuit topology of Figure 12 is designed with the intention of programming a single floating-gate transistor at a time with a focus on spatial efficiency. Decoders are used to select which row and column are connected to V_G and V_D, respectively, and are collocated

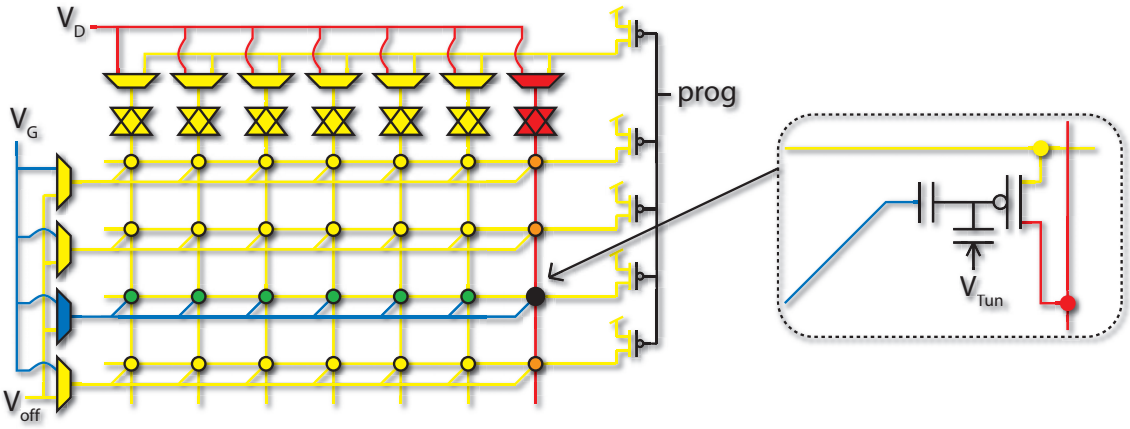


Figure 13. Floating-gate array isolation. In programming mode, all of the sources, drains, and gates are driven to V_{DD}, yellow, except a selected gate line, blue, and drain line, red. The intersection of the variable gate and drain line allow for a single floating-gate transistor to be selected. Other devices lack the V_{SG} or V_{SD} to be affected.

with the programming circuitry around the periphery of the array. The V_{off} voltage is used to reduce undesirable injection on rows that are unselected. The t-gates along the top of the array are necessary because in *run* mode, the drains should be connected to something other than the programming circuitry.

To program a single floating-gate transistor in the array, isolation of the desired device is important. To achieve the isolation of a single transistor, only a single row and column of the array are selected. Shown as the circled transistor in Figure 13, the intersection of the selected row and column provide the source, drain, and gate potentials necessary to form a channel. This is important because, as discussed in Section 2.2.2, a channel is necessary for DAHC injection.

3.2.1 Drain selection limitations

A potential problem of programming the fg-pFETs is that the current levels during programming become so great that a voltage drop across the drain-line t-gate is significant. The problem is illustrated schematically in Figure 14a. To evaluate the problem, a single, isolated, .5*um* floating-gate pFET was injected over a period of about 40 seconds with a V_{SD} of 5.5V. This V_{SD} was chosen to provide a measurable injection current. The device was

tunneled and then injected with a discrete 10 $k\Omega$ resistor placed between the drain node and the voltage source for the drain. The experiment was repeated for a 20 $k\Omega$ resistor. The resistors represent the possible worst-case parasitic resistance because of a transmission gate. The addition of the resistors resulted in a lower drain current, implying that the injection was limited. From the injection equation [21],

$$I_{inj} = I_{inj0}\left(\frac{I_S}{I_{S0}}\right)^{\alpha} e^{-V_{SD}/V_{inj}}, \tag{7}$$

it is shown that the injection is in part controlled by V_{SD}. As the current through the drain increases, the voltage dropped across the resistor increases. At 25 uA, the drop across the resistors is nearly 5-10% of the V_{SD} voltage for the 10 $k\Omega$ and 20 $k\Omega$ resistors, respectively. With a 5-10% decrease in the V_{SD}, the rate of injection is expected to decrease exponentially.

3.2.2 Parasitic Charge Movement

One other problem with the array structure of Figure 13 is that the selected element is not the only fg-pFET that will have its charge varied. In fact, the entire column of the selected transistor will experience some level of charge movement because of the high field at the drains of the transistors. FN tunneling from gate to drain and PN-junction reverse bias current and GIDL current will all cause movements in the floating-gate charge.

The channel hot electron (CHE) injection and FN tunneling are a subset of processes that belong to the broad field of leakage current mechanisms studied in the context of both non-volatile semiconductor and CMOS scalability and reliability [20][22]. The parasitic charge mechanism directly addressed in typical floating-gate array isolation is subthreshold conduction. Figure 6b is appropriate to discuss parasitic CHE injection, the only difference from intentional injection is that the channel potential is more positive, decreasing the availability of minority carriers for generating an impact-ionization event. Two mechanisms not addressed by typical array isolation are parasitic FN tunneling and PN-junction reverse bias current.

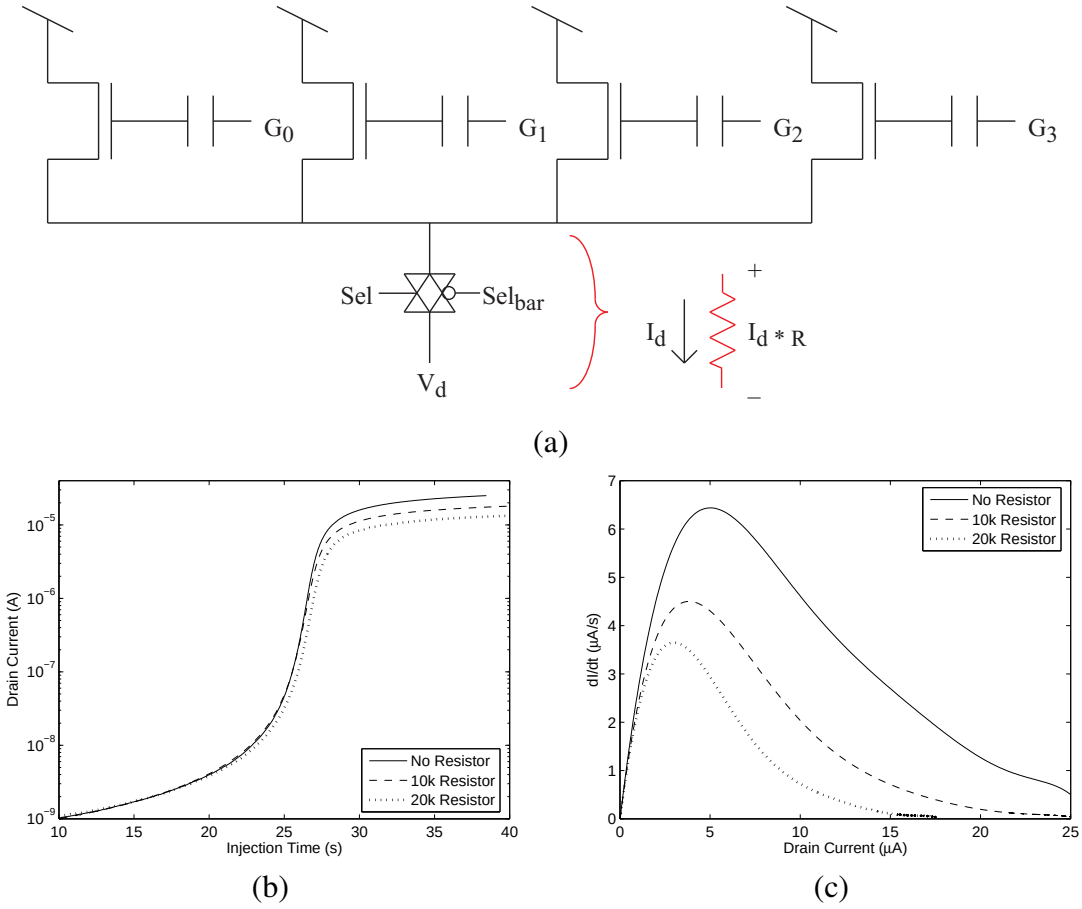


Figure 14. Illustration of drain resistor limitation. (a) Schematic drawing of the drain switch resistance. (b) Current measurement of injection for different discrete resistors. (c) Numerical temporal derivative of current versus current.

A field acting on the gate overlap between the drain and gate can be sufficient to cause FN tunneling. The voltage across the gate oxide is raised to the point electrons pass through the narrowed gate insulator onto the floating-gate. Shown in Figure 7, the condition arises because the gate is brought to the highest potential possible in order to minimize subthreshold conduction on columns where the applied drain voltage is near ground.

The PN-junction isolating the p+ drain region from the n-well of an unselected fg-pFET is subjected to a high field when another device on the same column has been selected for injection. Any thermally generated electrons that are swept through the high-field have the potential to inject into the gate. Exacerbating the situation is the gate overlap of the PN-junction at the inside edge of the drain. The high potential on the gate necessary to reduce subthreshold conduction can deplete or even invert the p+ region under the gate overlap. The severe band bending that occurs leads to band-to-band tunneling of electrons from the valence band to the conduction band. The current, called gate induced drain leakage (GIDL) current, has the potential to inject into the floating-gate, Figure 7.

3.2.2.1 Parasitic Charge Measurement

One of the more fundamental problems with characterizing a floating-gate transistor is that the charge parameter is difficult to measure directly. Amplifiers and capacitors can be used to integrate the current flowing onto the floating-gate, and transistors without floating-gates can be measured directly for gate current. Both methods require specific characterization structures that do not directly represent the exact conditions of the fg-pFET in the final working system. However, it is possible to directly measure a value proportional to the change in charge over time from a measurement available in a typical fg-pFET array, the gate sweep of a drain current. The accuracy of the charge measurement is limited by the estimation of capacitor through which the gate was swept, but is not strictly necessary for analysis of the charge rate since the capacitor is a constant (in the case of poly-poly capacitor).

3.2.2.2 *Temporal Derivative of Charge From the Drain Current*

The procedure of charge variation measurement proceeds with a subthreshold drain current I_d defined as:

$$I_d = I_{bias} \cdot e^{-\kappa V_{fg}/U_T} \tag{8}$$

where I_{bias} is the current that flows through the drain due to effects unrelated to the gate potential, κ is the coupling factor of the floating-gate to the surface potential, and U_T is the thermal voltage of a floating-gate transistor. The floating-gate voltage is then given by

$$V_{fg} = -\frac{U_T}{\kappa} \cdot \ln\left(\frac{I_d}{I_{bias}}\right) \tag{9}$$

The floating-gate voltage is not known, but it can be referenced to the known quantity V_g. By substituting (3) into (9), the primary gate coupling voltage is found to be

$$V_g = -\frac{C_T}{C_g}\left[\sum_j \frac{C_j}{C_T} \cdot V_j + \frac{Q}{C_T} + \frac{U_T}{\kappa} \cdot \ln\left(\frac{I_d}{I_{bias}}\right)\right] \tag{10}$$

where C_g is the primary capacitor though which a voltage is coupled to the floating-gate and V_j is the j^{th} voltage connected through capacitor C_j (which does not include V_g and C_g). The key observation about (10) is that by fixing the current and coupling voltages, the relationship between V_g and Q is uniquely defined, assuming a fixed temperature. By taking the temporal derivative of V_g for a fixed current I_d and coupling voltages V_j, the temporal derivative of charge becomes apparent.

$$\left.\frac{dV_g}{dt}\right|_{\substack{I_d \to fixed \\ V_j \to fixed}} = -\frac{C_T}{C_g} \cdot \frac{d\,(Q/C_T)}{dt} = -\frac{1}{C_g} \cdot \frac{dQ}{dt} \tag{11}$$

Experimentally, the meaning of quantity V_g for a fixed drain current is the V_g applied to cause I_d to flow. I_d is chosen such that (8) is valid, a condition satisfied by picking a subthreshold current. A desirable side effect of a fixed drain current and fixed coupling potentials is that κ is held fixed. When the V_g necessary to achieve I_d changes due to the

exposure of a high field, the new V_g along with the time of exposure results in a rate defined by (11). V_j is held fixed by performing the gate sweep with the same source, drain, bulk, and tunnel potentials applied. Thus, charge-movement can be measured from gate sweeps of a drain current.

3.2.2.3 Experiment

In order to explore the parasitic charge movement, the device under observation is first programmed to a particular charge level using a method similar to [23]. The drain current is then measured by sweeping the control gate under the conditions expressed in Figure 15a. Next, the device is exposed to a high field at the drain under a bias associated with no measurable drain current for a well defined period of time, illustrated in Figure 15b.

After high-field exposure, the device is once again measured under the condition of Figure 15a. For subsequent exposure times, the device is tunneled and injected back to the initial charge condition. The result of three exposures, 20, 40, and 60 seconds each, is plotted in Figure 15c. The experiment of Figure 15c is repeated for different initial charge levels in order to create a data set that spans the usable range of a fg-pFET with a $2.5V$ supply. A sub-portion of the complete data set is provided in Figure 16a. The individual experiments represented in Figure 15c are identifiable in Figure 16a by lines with a common hue.

In order to implement the fixed current condition of (11), a deep subthreshold current is chosen. The dashed line through $1nA$ on the plot of Figure 16a intersects a set of control gate voltages which serve as the experimental representation of (10). By subtracting the final control voltage $V_g\big|_{t>0}$ from the initial condition $V_g\big|_{t=0}$ and dividing by the associated time, a numerical result for (11) is found. Figure 16b is the result of computing the numerical derivative from Figure 16a. It is important to note that because charge and control gate voltage of Figure 16b have a one-to-one mapping, the value of charge must be lower for higher gate voltages. As a result, the effective floating-gate voltage during the high field exposure decreases for increasing control gate voltage on the plot.

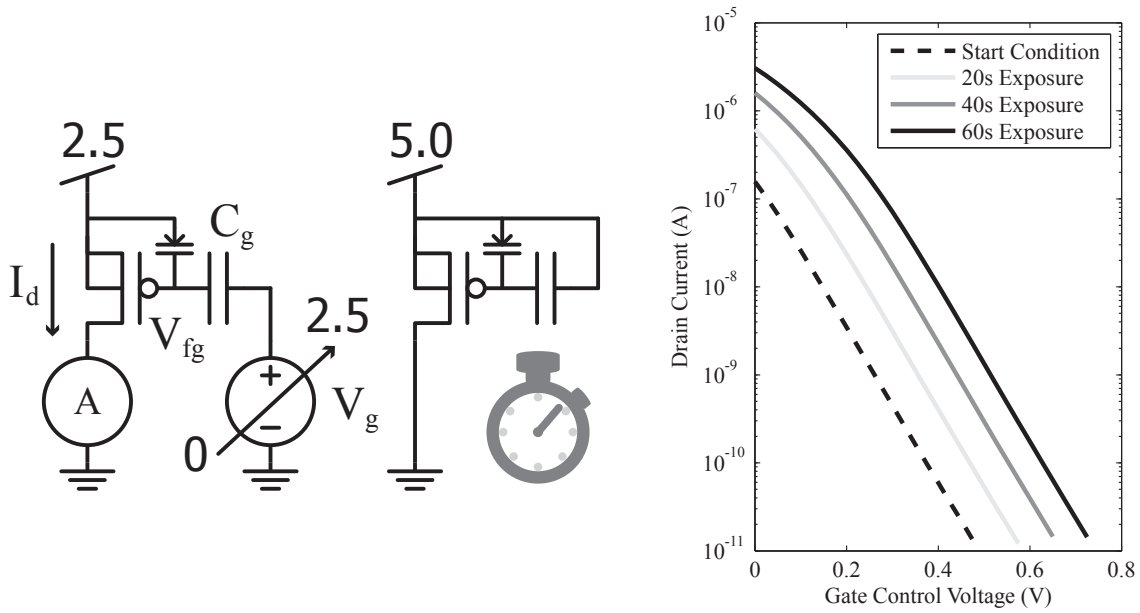


Figure 15. Experimental setup and procedure for measuring parasitic injection. (a) Circuit conditions used to perform a sweep of the floating-gate transistor. (b) Circuit conditions used to create the parasitic injection. (c) Experimental results demonstrating the effect of parasitic injection. Each sweep was performed under the circuit conditions in Figure 15a. The solid lines resulted from programming the floating-gate transistor to the dashed line then exposing the floating-gate to the circuit conditions in Figure 15b for 20 to 60 seconds.

For control gate voltages upwards of $2V$, subthreshold conduction injection dominates the rate of change of charge as predicted by (12). Below $2V$, $\frac{dQ}{dt}$ due to subthreshold conduction will continue to decrease exponentially. However, the subthreshold conduction effect becomes insignificant when compared to the effects due to the PN-junction reverse bias and tunneling currents.

3.2.2.4 Implications of Parasitic Charge Movement

The fundamental limitations imposed on floating-gate array injection by parasitic charge movement can be addressed through the bias point of the fg-pFET, the algorithm used for injection, and hardware used for isolation. The point on Figure 16b where $\frac{dQ}{dt}$ changes sign represents the bias point of minimum parasitic charge movement. fg-pFET arrays are often constructed with the bias point as a degree of freedom and could take advantage of that choice to minimize parasitic charge effects.

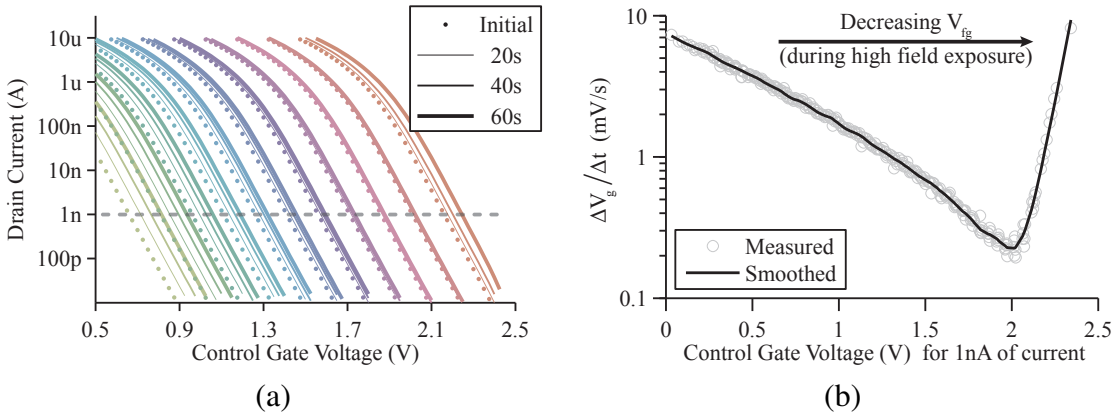


Figure 16. Parasitic charge movement data. (a) Representative sample of the raw data used to calculate $\frac{\Delta V_g}{\Delta t}$, which approximates $-\frac{1}{C_g}\frac{dQ}{dt}$ as shown in (11). Each color band is a repeat of the experiment in Figure 15 for a particular initial programmed charge. The line through $1nA$ is used as the reference for evaluating changes in V_g. (b) Rate calculated from the data in Figure 16a. The raw calculation, gray circles, is smoothed in order to view the trend, the solid black line. For higher V_g values with a fixed current, values of charge must be more negative. Thus, a lower effective floating-gate potential during high field exposure is present for higher control gate voltages in the plot. The charge movement on the left side of the graph is dominated by FN tunneling and gated PN-junction reverse bias current, while the charge movement to the right is dominated by subthreshold conduction.

When it is unfeasible to minimize the parasitic charge movement through bias point selection, or if the minimal charge movement still limits programming accuracy, the programming methodology provides a degree of freedom in minimizing the error due to charge variation. The parasitic charge movement has a strong time dependence, so by injecting each device on a shared column to a fraction of the desired final charge, the worst case charge movement will be much less than if all the devices were injected to their final value in succession. The fg-pFETs can be injected such that the final value is approached with smaller and smaller charge injection steps. The algorithm described in [24] takes such an approach. The work does not explicitly address parasitic charge movement, but provides an approach that is insensitive to errors in the charge injection model through a multi-inject and measure approach. Injection steps can be traded off for increased accuracy in order to address the unmodeled charge movement. It is also possible to use the data set in Figure 16b to predict the parasitic charge variation from injecting a column of devices and incorporate it directly into the injection routine, potentially nullifying the impact of parasitic

charge movement.

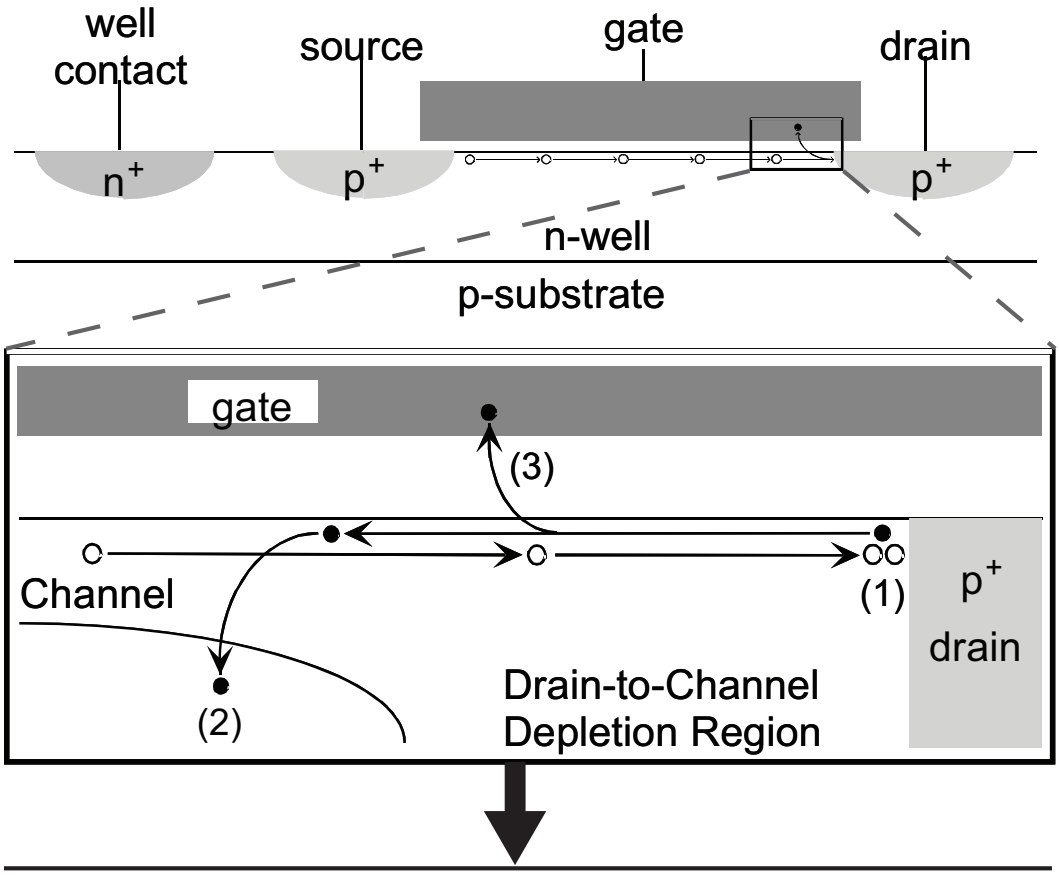
In order to reduce parasitic charge movement during array injection to a negligible level, it is necessary to prevent the high field from forming around unselected devices. The most direct way to modify the array element would be to place a switch on the drain of the fg-pFET in order to shield it from the injection pulse. In [25], the use of additional selection circuitry is discussed in the context of array isolation, and switches on either the source or drain of a fg-pFET are considered. The findings of that work can be augmented in light of the data herein. Specifically, a switch on the source of an array element limits parasitic charge movement for the range of biases where subthreshold conduction dominates, as the source of carriers necessary for CHE injection are minimized. A switch on the drain minimizes parasitic charge movement over the entire usable range of the device by preventing the parasitic field at the drain from forming.

CHAPTER 4

FG-PFET SIMULATION

Large-scale analog reconfigurable systems are enabled by analog memory. Floating-gate charge storage, a powerful analog memory technique, is reaching a critical mass of understanding and usage. Research VLSI systems have been fabricated and demonstrated using the floating-gate pFETs (fg-pFETs) for a myriad of applications. Leveraging these large-scale systems requires attention to the programming approach, which is critical to the system behavior and use. One approach, employed by the FPAA discussed in the previous chapter, uses a rigorous characterization method and an inject/measure routine to inject the fg-pFETs [24]. It is useful to think of such an approach as being global-feedback intensive; a supervisor system external to the IC is used to program the device. The pattern classifier of [17] takes advantage of local-feedback to reduce the need for external measurement and processing, as described in [26]. Both methods of programming are appropriate under certain conditions, and there a range of other programming methodologies that use varying levels of local and global feedback. However, the trade-offs are not immediately obvious. The use of a programming methodology in a given system often requires a fabrication step to properly investigate how effective the approach is for the system. Thus, in order for floating-gate injection to become a ubiquitous analog design methodology, as opposed to a risky technique, a robust model with strong simulation support is necessary.

In this section, I describe floating-gate injection, along with examples of simulator-targeted floating-gate injection models. Next, I extend one of the models, apply it to Verilog-A, captured succinctly in Figure 17, and demonstrate how to use the drain current of a floating-gate to fit the model for simulation.




```
1.  if( Vsd > Vsd_inj ) begin
2.    Iinj=Iinj0*Is/Ith*
3.         exp(-kappa/Vinj*(Vb-Vg-(Vb0-Vg0)))*
4.         exp(     1/Vinj*(Vb-Vd-(Vb0-Vd0)));
5.  end else begin
6.    Iinj=0;
7.  end
```

Figure 17. Simplified process of channel hot-electron injection in pFET and the associated Verilog-A model description for floating-gate injection simulation. Minority carriers are accelerated to the point of impact ionization, event (1), and the resulting electron is swept through the channel toward either the well or the oxide. The rate of electrons reaching the gate, event (3), compose the injection current. The actual Verilog-A code used for simulating floating-gate injection is provided. The model has gate, source, drain, bulk, and source current ports. For the sake of readability, potentials like V(source,drain) were assigned to variables such as Vsd. The value I_{inj} is applied as a current between the drain and gate.

4.1 Channel Hot-Electron Injection

A floating-gate pFET is a pFET with an electrically isolated gate. By adding and removing charge from the gate, the I-V relationship of the transistor can modified. We use channel hot-electron (CHE) injection to add charge to the gate and Fowler-Nordheim tunneling to remove charge. Because we treat tunneling as a global erase, we only need to model CHE injection to enable our subset of floating-gate system design.

CHE injection is a multiple-step process where a high-energy hole results in an additional electron on the floating-gate pFET, Figure 17. When a hole is sufficiently accelerated from source to drain, it can impact ionize, resulting in two holes and an electron at the boundary of the drain region. The resulting electron will travel back towards the channel, accelerated through the same field that energized the hole. If the energy imparted on the electron is sufficient and the field between channel and gate is in the right direction, the electron will travel through the oxide onto the gate.

CHE injection is dependent on the source current and drain-to-channel potential (Φ_{dc}). However, Φ_{dc} is not explicitly available in a typical circuit simulation. [27] presents a first-order model of CHE valid for subthreshold and above-threshold injection that does not depend on Φ_{dc}:

$$I_{inj} = I_{inj0}\left(\frac{I_s}{I_{th}}e^{-\kappa\frac{\Delta V_{fg}}{V_{inj}}}\right)e^{-\frac{\Delta V_{ds}}{V_{inj}}}, \tag{12}$$

where κ is the coupling coefficient from the floating-gate to the channel; V_{inj} is a device parameter dependent on the biasing of the drain-to-channel potential; I_{inj0} is the bias injection current flowing when the transistor is biased at the threshold current, I_{th}; ΔV_{fg} is the change in the floating-gate voltage; $\Delta V_{ds}$is the change in the drain-source potential; and I_s is the source current. Another approach, presented in [28], also provides a model independent of Φ_{dc}. However, it is an empirical model and requires the oxide current in order to fit the model, a difficult data set to obtain.

The model expressed in [27] is desirable because it is physically based and easily implemented as a macro model in a circuit simulator. But, because there is no bulk reference, the parameters are bias dependent. For instance, I_{inj0} is only valid for a particular initial floating-gate voltage, V_{fg0}. Further, by referencing the drain to the source, the model will under-predict the injection current when the source potential drops and all the other potentials are fixed. Both issues can be remedied by using the bulk, V_b, as a reference and eliminating the source potential:

$$I_{inj} = I_{inj0}\left(\frac{I_s}{I_{th}}e^{-\kappa\frac{\Delta\left(V_b-V_{fg}\right)}{V_{inj}}}\right)e^{-\frac{\Delta(V_b-V_d)}{V_{inj}}}. \tag{13}$$

The modification is critical for model implementation because the bias point for injection characterization is often different than the operational bias point of the transistor. We require the flexibility in the model so that it can simulate the device correctly and facilitate insights in as many situations as possible.

4.2 Modeling and Simulation

Verilog-A was used to model (13) because Verilog-A is a continuous-time modeling language that integrates easily into the Cadence Spectre and it allows for compact, human-readable modeling. The computationally significant portion of the Verilog-A model implemented for this paper is shown in Figure 17. As implied by the code, the model takes as inputs the source, drain, floating-gate, and bulk potential, as well as the source current; all of the other terms are set as parameters. The if-statement prevents the injection model from interacting with the simulator during the DC-point calculation. A DC source with zero potential is used to monitor the source current and transmit it to the model input. A representation of the schematic for a floating-gate simulation is shown in Figure 18.

In order to properly simulate the floating-gate transistor, the schematic must first be *programmed* in order provide the simulator with a DC starting point for the floating-node. This is accomplished with a large resistor between a high-gain voltage-controlled voltage source (vcvs) in feedback around the transistor and the floating-node. The resistor is set to

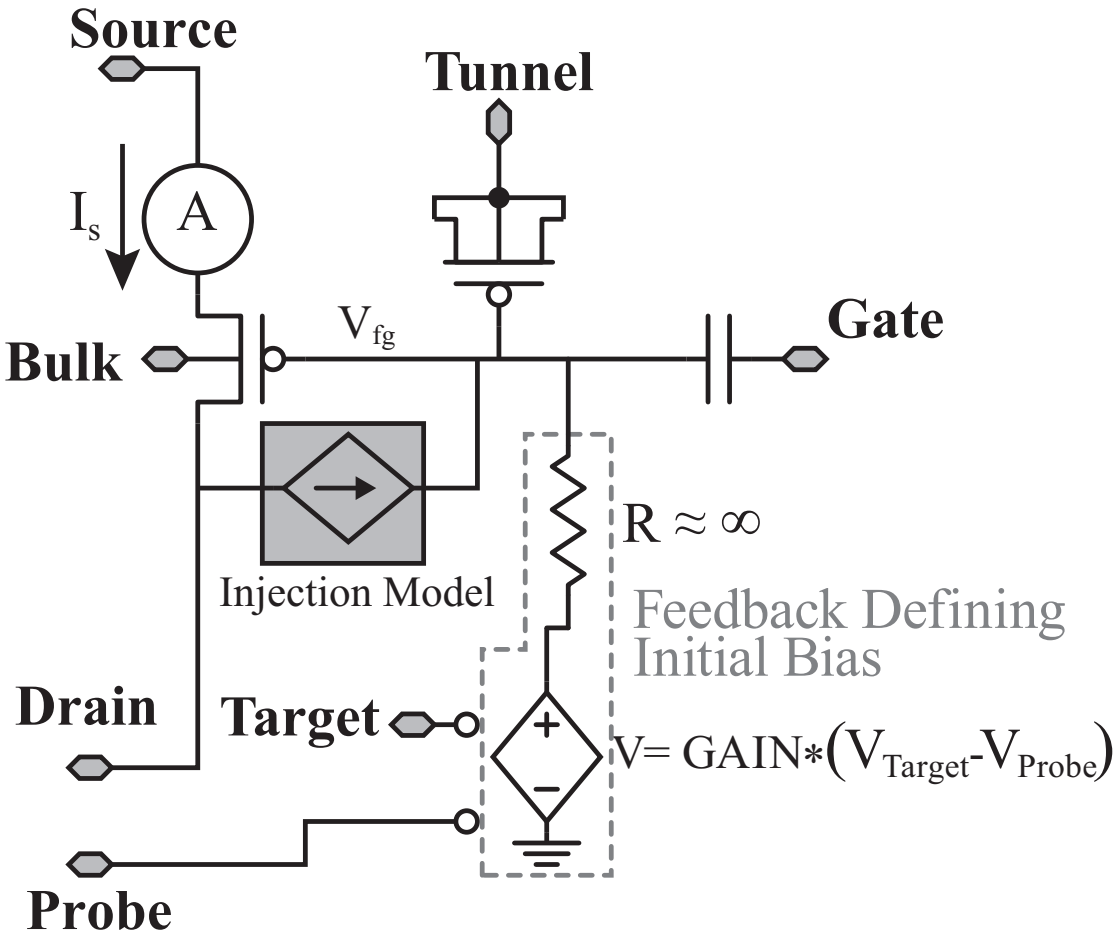


Figure 18. A representation of the schematic used for simulation. Cadence Spectre was used to integrate the design-kit transistor models with the Verilog-A implementation of injection. The voltage-controlled voltage source (vcvs) forms the negative feedback loop for setting the initial *charge* of the fg-pFET. The result of this circuit was used to generate the simulation data in Figures 19 and 20.

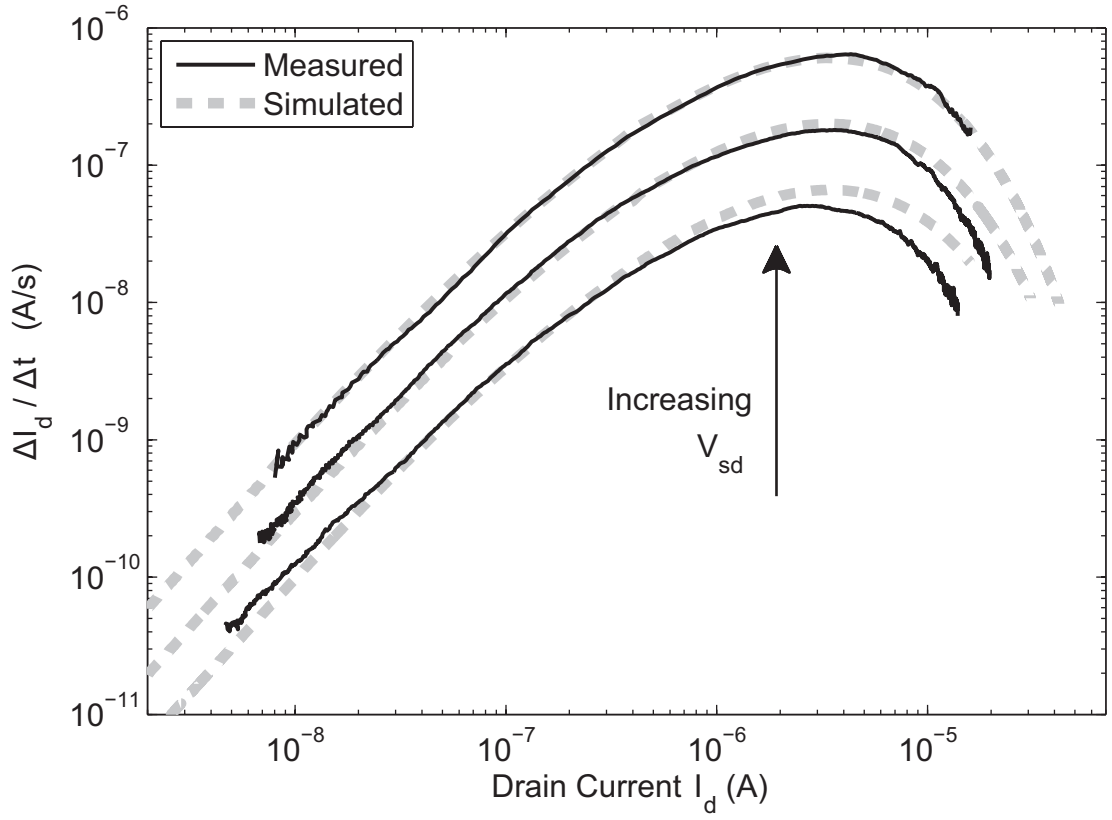


Figure 19. Measured experimental and simulation data of $\frac{dI_d}{dI_t}$ vs I_d. The drain current of a 0.35um fg-pFET with fixed gate, source, bulk, and three different drain potentials, $V_{sd} = \{3.7, 3.8, 3.9\}$, was measured, and the resulting numerical time derivative was used to fit the model in Figure 17. The result of the simulated data is superimposed on measured data, showing a good fit for the majority of the drain current. Because V_{inj} is bias dependent, some deviation is expected as V_{sd} changes. The data measured off of the real fg-pFET required smoothing because the noise from the off-chip current measurement combined with the numerical derivative made the plot difficult to read otherwise.

the maximum value supported by the simulator so that it only impacts the DC solution of the simulation. The current monitor is used once again to transmit the value of the bias current to the vcvs, and thus the DC solution will converge on the floating-gate voltage necessary to achieve a particular DC current. The DC simulation cannot be used for DC sweeps, however, because the external potentials coupling through capacitors will be resolved to open circuits. Accordingly, a transient simulation is used in order to correctly calculate floating-gate coupling. In order to correctly balance the floating-gate "programmed" voltage, the transient simulation must start all potentials from the DC solution.

The parameters of the injection model were fit against a fabricated floating gate of an equally drawn size. In this case, a 1.8um/.8um floating-gate transistor in a .35um process was used to extract κ and I_{th}. Next, the transistor was tunneled and then injected under a fixed bias while measuring the current. The experiment was performed for three different drain voltages. The fabricated fg-pFET used for experimental measurements was not instrumented to measure I_{inj}, so an alternative relationship for I_{inj} characterization was used. All of the charge change in Figure 19 is due to I_{inj}, which means the time derivative of source current is proportional to the injection current, as given by

$$\frac{dI_s}{dt} = \frac{dI_s}{dQ} \cdot \frac{dQ}{dt} = \frac{dI_s}{dQ} \cdot I_{inj} \tag{14}$$

By comparing $\frac{dI_s}{dt}$ values with a constant I_s, the variation in $\frac{dI_s}{dt}$ is entirely due to variations in I_{inj}. A numerical approximation of (14) was calculated and plotted against current, displayed in Figure 19. Because the measured fg-pFET did not have the source pinned out, the drain current was used in its place.

V_{inj} was extracted by examining $\frac{dI_d}{dt}$ for fixed I_d; the ratios of $\frac{dI_d}{dt}$ among the different V_{sd} experiments are used to solve for V_{inj} in (13). A simulation was run with the extracted parameters, κ was tweaked to fit the measured curvature of the plot, and I_{inj0} was used to scale the $\frac{dI_d}{dt}$ curve to the appropriate level. The dashed lines represent the simulated results of the parameter extraction. Once the $\frac{dI_d}{dt}$ plots were matched, the transient progression of the source current was used to verify the model, Figure 20.

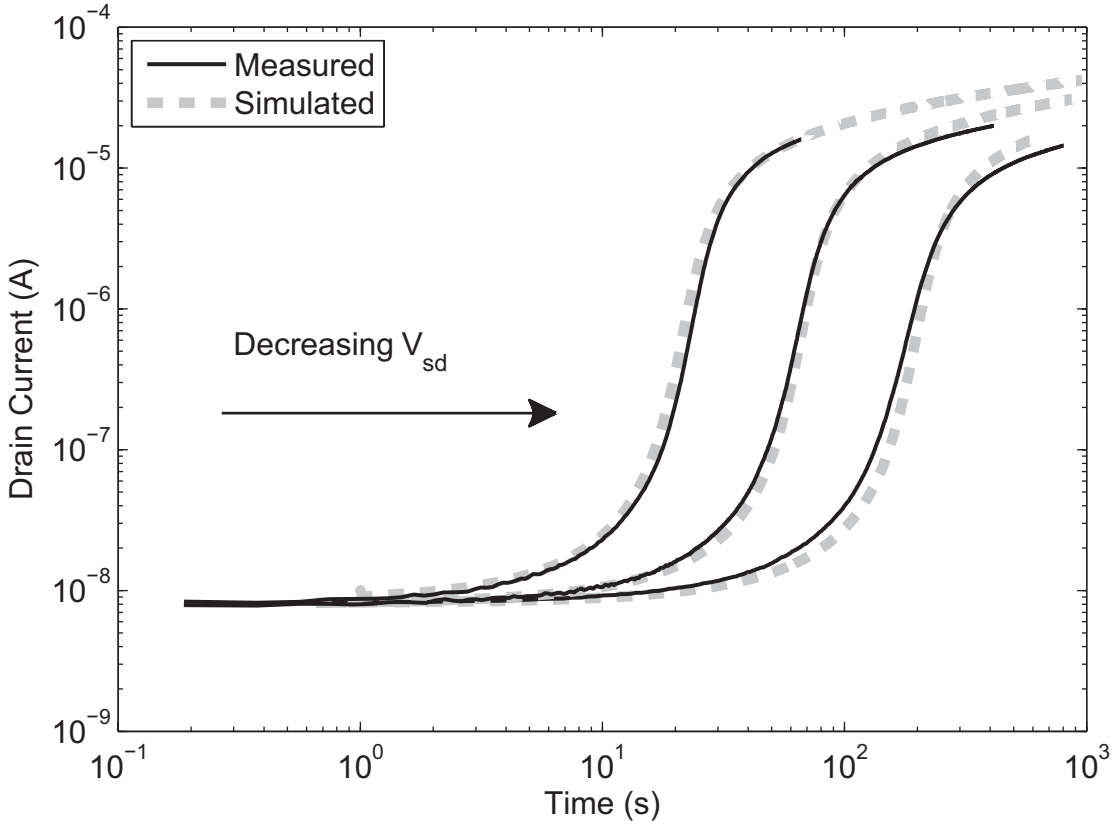


Figure 20. The raw measured experimental and simulation data used to generate the curves in Figure 19. The source and bulk were held at 4.4 V, and the gate was set to produce a current of about 8nA with the drain at 0.7, 0.6, and 0.5 volts. The floating-gate transistor used for modeling was from the same IC as was used by [27], the FPAA in [19].

CHAPTER 5

VECTOR MATRIX MULTIPLICATION CELL

We live in an analog world. By that I mean the sounds we hear and the light we see are apparent to us as a continuum. And as long as we limit our discussion to the signals of significant magnitude where quanta of energy are indistinguishable from one another, the signals we measure are also analog in nature. It is then worth considering why the majority of processing that we perform on analog signals takes place in digital signal processors (DSPs). DSPs are so abundantly consumed because they provide a straightforward mapping between mathematical descriptions and physical implementations of algorithms. But between the costly analog to digital conversions required and the crutch of 10's of bits of resolution, DSPs are no panacea for constrained computation. In [29], Sarpeshkar showed how for a fixed power or area consumption, analog computation is more efficient than digital computation for an output SNR less than about 10 bits. The crutch of 10's of bits of resolution is a reference to a design environment where the availability of the resolution results in designers going down algorithmic paths which lead to solutions far to the right of the flicker noise limit of analog. If the algorithms had been designed with an SNR constraint under 10 bits, the resulting implementation would not only be more efficient if it were mapped to analog, but for a fixed area, the power consumption would decrease exponentially with respect to the digital implementation as the SNR requirement was reduced. My point here is the same as the point I make in the outset of this document, that engineering efficient computational systems are dependent on good analogies between math and implementation. And if our goal is to replace digital signal processing with analog signal processing, then a useful place to start is with the core element of a DSP, the multiply-accumulate.

A multiply-accumulate operation is as it sounds, a multiplication of two numbers that

is added to an accumulator, and it is the workhorse of DSP computation. In analog, storage is costly, but summation is not, particularly when the signals are represented by a current. Kirkov's current law describes how currents that are wired together are summed together. So in analog, accumulation tends to be most efficiently implemented though parallelism, provided that all the signals associated with multiplication are available at once. By combining programmable current mirrors with KCL, we can approximate a multiply-accumulate operation.

5.1 Programmable Current Mirror

As for implementing multiplication, lets consider the case where we use KCL. If you had a signal that required some integer multiplication m, you could replicate your signal m times. And then for division we could shift the representation of $1x$ to mean some integer multiple, and division could be done using so many signal replications less than $1x$, and such a system would be painfully unwieldy. The majority of the time we use relative geometries of transistors in place of KCL to implement a multiplication. Take the reduced form of the EKV model of an NMOS transistor in saturation,

$$I_d = \frac{W}{L} I_s \ln^2\left(1 + e^{\frac{\kappa(V_g - |Vt|) - V_s}{2U_t}}\right) \tag{15}$$

If the drain current is taken as an output for a fixed source and gate voltage, the output will change with the ratio of the width to length of the transistor. The current mirror, a strong contender for the most common analog circuit, implements a multiplication with a weight proportional to the ratio of two transistors geometries. For the current mirror in Figure 21a, $I_{out} = \frac{(W/L)_2}{(W/L)_1} I_{in}$, provided all of the device parameters are matched and the two devices are operating in the same regime.

Because our goal is efficient computation, its worth considering the two regimes implied by (15), weak inversion and strong inversion. By virtue of implementing a multiplication, the input and output transistors will likely have different ranges of operation. If over the range of operation, one of the transistors moves between weak and strong inversion, the

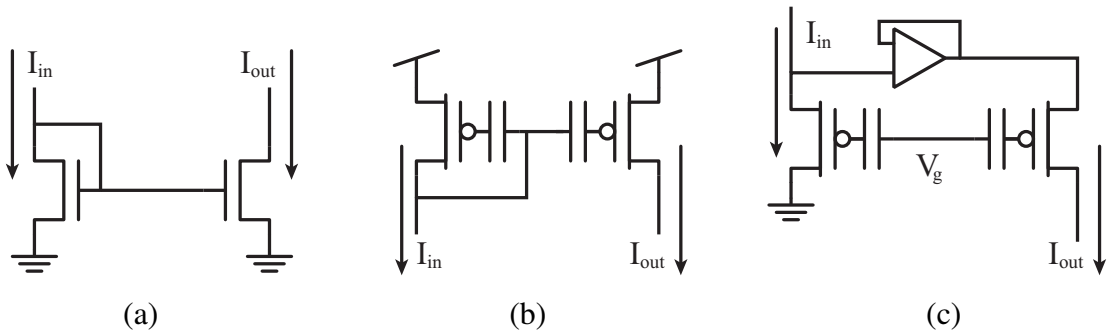


Figure 21. Source and gate programmable gain current mirrors. (a) Basic current mirror. The multiplication is fixed by the geometries of the input and output transistors. The linearity of the multiplication is limited by the threshold current and the ratio of U_t, and κ, assuming a fixed drain voltage. (b) Floating-gate current mirror. The multiplication is controlled using floating-gate programming. The linearity of the multiplication has an additional dependence on the ratios of $\frac{C_g}{C_T}$ between the input and output transistor. (c) Floating-gate current mirror using the source. Unlike the gate-mirroring, the linearity of the multiplication no longer depends on κ and the other capacitors on the floating node. However, the gain of the amplifier will limit linearity.

resultant multiplication will vary with some non-linearity related to the movement between e^x and x^2. While that specific non-linearity could be advantageous, it is not helpful for implementing the linear operations of a multiply-accumulate cell. In terms of power, the biases associated with strong inversion will be undesirable if the intent is to implement massively parallel multipliers. If we consider efficiency in terms the amount of transconductance a transistor produces for a given bias current, we get a result that looks like Figure 22. Within the subthreshold regime, $\frac{gm}{I}$ is a fixed quantity, $\frac{\kappa}{U_T}$, and the maximum for the transistor. In the above-threshold regime, $\frac{gm}{I}$ decreases at a rate of $\frac{1}{\sqrt{I}}$. In addition, the exponential function available in subthreshold operation can be harnessed for log-domain computations, lending a strong mapping to the floating-point work done in the digital domain, and making high dynamic range computations more approachable in analog. Finally, $ln(x)$ and $e^{(x)}$ are expensive operations in the digital domain. In analog, such operations are accessible directly from a single transistor, further lending to the desirability of weak inversion.

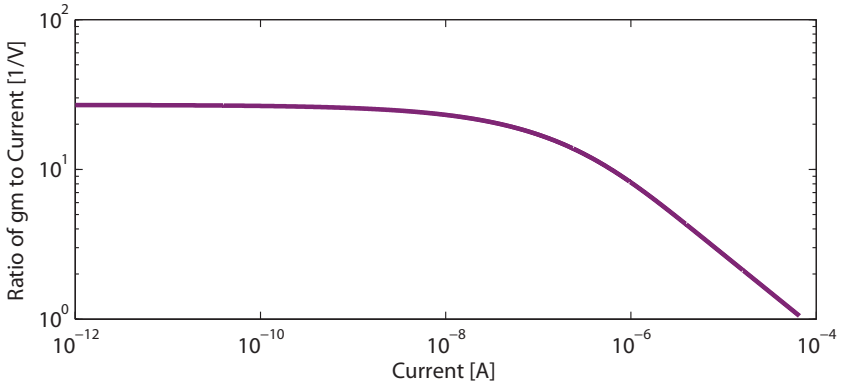


Figure 22. Ratio of transconductance to current. The data was taken from a simplified EKV model. Within the subthreshold regime, $\frac{gm}{I}$ is a fixed quantity, $\frac{\kappa}{U_T}$, and a maximum for the transistor. In the above-threshold regime, $\frac{gm}{I}$ decreases at a rate of $\frac{1}{\sqrt{I}}$.

By constraining ourselves to subthreshold, (15) reduces to

$$I_d = I_0 \, e^{\frac{\kappa V_g - V_s}{U_t}} \tag{16}$$

where W/L and $e^{\frac{\kappa}{U_t}|Vt|}$ were included in I_0. For the circuit in Figure 21a,

$$I_{out} = I_{0_{out}} \, e^{\left(\frac{\kappa_{out}}{\kappa_{in}} \frac{U_{t_{in}}}{U_{t_{out}}} \ln\left(\frac{I_{in}}{I_{0_{in}}}\right)\right)} = m \, I_{in}^{\alpha}; \; \alpha = \frac{\kappa_{out}}{\kappa_{in}} \frac{U_{t_{in}}}{U_{t_{out}}} \; ; \; m = \frac{I_{0_{out}}}{I_{0_{in}}^{\alpha}} \tag{17}$$

For $\alpha = 1$, the multiplication m is set by the ratio of I_0's. Such a multiplication is sensitive to variances in manufacturing process. The sensitivity is addressed through increasing the absolute areas of the transistors involved and distributing the transistors in such a way as to desensitize the macroscopic devices to processing gradients, common centroid layout. But we are interested in approaching the capability of a MAC unit on DSP, which suggests that a fixed multiplication will not suffice; it is not feasible to incorporate a continuum of mirrors on a chip in order to implement the continuum of potential multiplications.

By using the floating-gate current mirror in Figure 21b, we can simultaneously address some of the process variances and the need for a continuum of multiplications. First, consider the drain current of (16), re-written for a pFET with the floating-gate coupling and charge in mind:

$$I_d = I_0 \, e^{\frac{-\kappa}{U_t}\left(\frac{C_g}{C_T} V_g + \frac{Q}{C_T}\right)} \tag{18}$$

When we determine the output current, we find

$$I_{out} = m\, I_{in}^{\alpha};\ \alpha = \left(\frac{U_t C_T}{\kappa C_g}\right)_{in}\left(\frac{\kappa C_g}{U_t C_T}\right)_{out};\ m = \frac{I_{0_{out}}}{I_{0_{in}}^{\alpha}}\, e^{\beta\left(\frac{\kappa}{U_t}\right)_{out}}; \beta = \left(\frac{C_{g_{out}}}{C_{T_{out}}}\frac{Q_{in}}{C_{g_{in}}} - \frac{Q_{out}}{C_{T_{out}}}\right) \tag{19}$$

If we again examine the scenario where $\alpha = 1$, we see a multiplication from the ratio of I_0's. But more importantly, we also have a multiplication term $e^{\beta\left(\frac{\kappa}{U_t}\right)_{out}}$ which is sensitive to floating-gate programming. As a result, precise multiplications are possible through correctly controlling the charge on the floating-nodes.

The current mirror is composed of two components, a front-end circuit that computes a log-compressed version of the current and a back-end circuit that implements a log-domain addition resulting in a multiplication of current. By broadcasting the log-compressed front-end voltages across an array, the programmable current mirror becomes a programmable vector-matrix multiplier (VMM), [30]. Shown in Figure 23a, the current out of the j^{th} column is, by KCL, the summation of the currents dependent on the vector of log-compressed voltages computed from the vector of input currents.

$$I_j = \sum_{i=1}^{m} I_{i,j};\ I_{i,j} = I_{b_{i,j}}\, e^{-\frac{\kappa_{eff_{i,j}}}{U_{t_{i,j}}} V_i};\ V_i = -\ln\left(\frac{I_i}{I_{b_i}}\right)\frac{\kappa_{eff_i}}{U_{t_i}} \tag{20}$$

where the charge term has been included in I_b and $\kappa_{eff} = \kappa\frac{C_g}{C_T}$.

Up to this point, we've only considered a value of one for α. However for large dynamic range signals, the actual value of is α is critical for the mirrors and VMM of Figures 21a and 21b. In particular, α limits the linearity of the multiplications because it represents a power law. For a fixed, non-zero α, the error in the multiplication for a change in the input current by a factor of r is:

$$\pm\ \%\ error = \left(r^{\alpha-1} - 1\right) * 50 \tag{21}$$

To put that in perspective, to keep the error within ±2.5% over two and a half decades requires an alpha of just less than 1.009. The choice of reference current is arbitrary, so when defining a multiplication it is desirable to choose a signal from the highest range if the effect of α is compressive ($\alpha > 1$) and a vice versa if α is expansive ($\alpha < 1$). For instance in the previous calculation, $r = 10^{-2.5}$. If α were less than one, we would use $r = 10^{2.5}$.

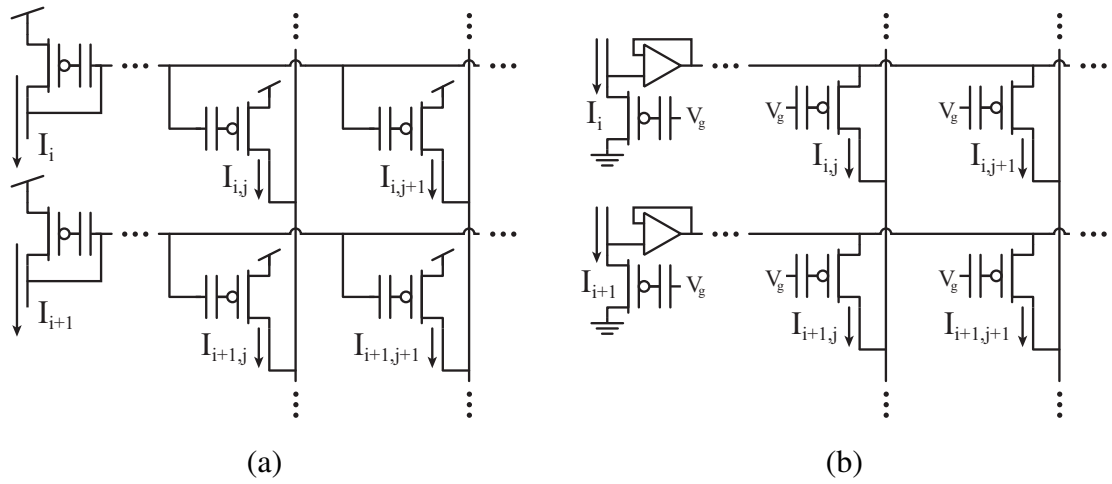


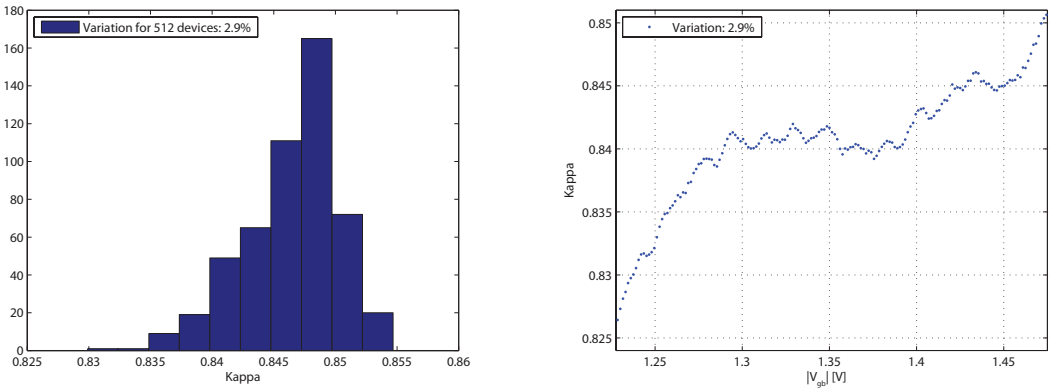
Figure 23. Source and gate vector matrix multiplication. (a) Vector matrix multiplier based on Figure 21b. (b) Vector matrix multiplier based on Figure21c.

As for what constitutes α, the thermal voltages of the input and output transistors are only a concern if there is a thermal gradient across the chip; under most circumstances the thermal voltage can be neglected. We can expect the ratio of the gate to total capacitance to have up to 1% error, based on the work done in [31]. Variation in κ is another concern. Figure 24a is an illustration of a κ measurement for 512 pFETs. The arrangement on the chip is that of a column input for a VMM. In that case there was a 3% variation across the devices. In Figure 24b the subthreshold slope of a single device is plotted, showing a nearly 3% change in κ for a gate voltage that corresponds with nearly three orders of magnitude of drain current.

There is an alternative to building a multiplication with such a strong dependence on the matching of components. Rather than using the gate for computing variable signals, we can use the source [32]. Illustrated in Figure 21c, source mirroring has the following relationship between output current, broadcast voltage, source voltage, and input current:

$$I_{out} = I_{0_{out}} e^{-\left(\frac{\kappa}{U_t} V_{fg}\right)_{out}} e^{\frac{V_{bc}}{U_{t_{out}}}} \; ; \; V_{bc} = -\left(\frac{A}{1+A}\right) V_s \; ; \; V_s = U_{t_{in}} \ln\left(\frac{I_{in}}{I_{0_{out}}}\right) + \kappa_{in} V_{fg_{in}} \tag{22}$$

where V_{bc} is the broadcast voltage and A is the openloop gain of the amplifier. (22) reduces



(a) (b)

Figure 24. Experimental measurement of κ mismatch. (a) Histogram of κ for 512 devices. (b) Kappa for different V_{bg} voltages.

to

$$I_{out} = m\, I_{in}^{\alpha};\ \alpha = A_{CL}\frac{U_{t_{in}}}{U_{t_{out}}};\ m = \frac{I_{0_{out}}}{I_{0_{in}}^{\alpha}}\, e^{\frac{\beta}{U_{t_{out}}}}\ \beta = A_{CL}\left(\kappa V_{fg}\right)_{in} - \left(\kappa V_{fg}\right)_{out} \tag{23}$$

where A_{CL} is the closed loop gain of the amplifier driving the source of the output transistor. The first important issue to note in (23) is that α no longer has a dependence on capacitor and κ ratios. The next is to recognize that aside from thermal mismatch, the only other element of the current-gain power law is A_{CL}, which can be designed to a desirable tolerance in a straightforward fashion. For instance, an openloop gain of 41 dB is enough for ±2.5% error over two and a half decades; with a modest openloop gain of 50 dB the power-law results in an error within ±2.5% over seven decades of input current.

Another key aspect of source mirroring is the relationship between the input current, the output current, and the floating-gate voltages. Changes on the source couple into the floating-gate, and that coupling will introduce an error by modifying the multiplication based on the signal change. In particular, care must be taken in sizing $\frac{C_{gs}}{C_T}$, the ratio of the gate-source capacitance to the total capacitance, as it controls the coupling. Source mirroring readily extends to a VMM [33], as shown in Figure 23b.

One final point about source mirroring is that it is nearly insensitive to κ variation. κ is defined as $\frac{C_{ox}}{C_{ox}+C_D}$, where C_{ox} is the transistor oxide capacitance between the gate and

channel while C_D is the depletion capacitance between the channel and the substrate. C_D is a MOS-capacitor phenomenon, and is weakly dependent on voltages other than the gate-to-bulk potential. As long as a particular multiplication is referenced to a particular input-output current ratio, then the multiplication will necessarily have taken into account the κ movement due to that particular multiplication.

5.2 Input and Output Terminals

The VMM discussed in the previous section is a current input, current output structure with the currents constrained to weak inversion. One of the critical aspects of a VMM implementation is the instrumentation of the input and output currents.

First, consider matter of the input and output terminals of the gate-mirrored VMM and the output of the source VMM. In order to hold the voltages at those terminals fixed, either the current must remain fixed, which defeats the purpose of the VMM, or an active element must be used. We can determine to the extent we need to fix the input and output drain voltages by using a model for the Early effect of the transistors. In subthreshold, a reasonable model for the Early effect is

$$I_d = I\, e^{\frac{V_{bd}}{V_A}} \tag{24}$$

where I is the drain current for a pFET without considering the Early effect, V_{bd} is the bulk to drain voltage, and V_A is the Early voltage. For a range of drain-to-bulk potentials of ΔV_{db}, the error in the current is

$$\%error = (e^{\frac{\Delta V_{db}}{V_A}} - 1) * 100 \tag{25}$$

It is not sufficient to neglect the Early effect on the basis having the same dimension transistor on the input and output, because the currents though the input and output transistors are necessarily different, except for multiplications of unity.

In Figure 21c, a transistor is used to convert the incoming current to a log-compressed

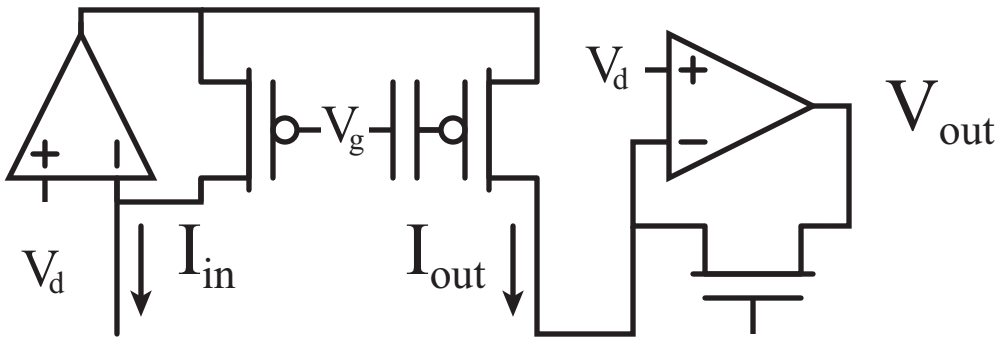


Figure 25. Transimpedance amplifiers in a source signaling current mirror. The amplifiers fix the input and output drain voltages while increasing then BW by the gain of the amplifiers, A.

voltage, and a follower is used to convey that voltage to the source of the output transistor. The bandwidth of that conveyance is set by the time constant of the input node. In particular,

$$\omega = \frac{1}{RC} = \frac{gm}{C} = \frac{\kappa}{U_t}\frac{I}{C} \tag{26}$$

To put that in perspective, $100pA$ into a $100fA$ will yield a bandwidth of about $4\,kHz$. A way to increase that frequency is to implement a transimpedance amplifier out of the sensing transistor, as shown in Figure 25. The effective input resistance of the structure will be $\frac{1}{A\,gm}$, which will increase the bandwidth by a factor of A. The same thing can be done for the gate-mirroring. A desirable side effect is that the amplifier will also reduce the variation in the drain node by a factor of A.

5.3 Frequency Response at the Gate

In designing the computation transistor to be tiled out in the VMM, there are several issues that must be kept in mind. One such issue is related to the coupling of the input signal from the source to the gate of the computational FET. For a particular bandwidth at the source of the transistor characterized by the frequency $f_{BW\,source}$ and a $3dB$ attenuation from unity, the transfer function from the source to gate of the computational FET must be $\frac{1}{2^n}$ times smaller at the gain at $f_{BW\,source}$ in order to be able to implement the multiplication with a

precision of n bits, which is $3 + 20\log_{10}(2^n)$ dB down from unity. In the case where the computational FET is implemented as a floating-gate transistor, that transfer function is defined as:

$$\frac{Vgate}{Vsource} = \frac{C_{gs}}{C_T} \tag{27}$$

where C_{gs} is the gate to source overlap capacitance and C_T is the total capacitance at the floating-gate node. As a result, C_T must be sized to as follows for an *n bit* precise multiplication:

$$C_T \geq C_{gs} \cdot 2^n \cdot 10^{3/20} \tag{28}$$

To put that in perspective, a $3fF$ overlap capacitor requires a $1pF$ capacitor for an 8-bit accurate multiplication.

However, the overlap capacitor is not a free parameter to reduce. If we consider the threshold current to be the maximum representable signal level that still produces weak-inversion characteristics, then by choosing a maximum desired current level we fix the W/L ratio, since $I_{th} = \frac{W}{L} I_{th_{unit}}$. The length is fixed by Early voltage, which leaves the width as a free parameter, which is proportional to C_{gs}. The way a maximum current level is chosen is by the limit imposed by the system bandwidth and the desired dynamic range. Assuming we use a transimpedance amplifier like we discussed from the previous section,

$$I_{min} = \frac{U_t}{\kappa}\frac{\omega C}{A} \tag{29}$$

The maximum current is then simply $DR * I_{min}$.

One way to design around the limit C_{gs} imposes on the size of the VMM cell is to use an amplifier to drive the computational FET instead of using a floating-gate transistor, Figure 26a. In this case, the transfer function from source to gate is

$$\frac{Vgate}{Vsource} = \frac{C_{gs}}{C_T}\frac{sRC_T}{1 + sRC_T} \tag{30}$$

By designing the amplifier such that the roll-off from the zero is $20 \cdot log10\,(2^{-n})\ dB$ below $f_{BWsource}$, the precision constraint is maintained. The separable transform imager discussed

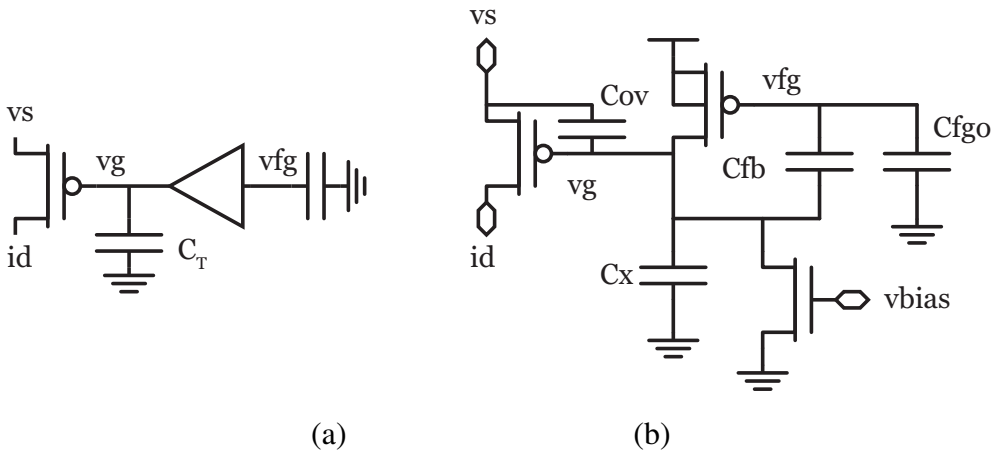


Figure 26. Floating-gate VMM cell with computation transistor gate driven by amplifier. (a) Conceptual drawing. (B) Implementation of Figure 26a. .

in Section 6.1 requires $f_{BWsource} = 208\,kHz$ in order to achieve $32\,fps$ for a 5τ settling time. In that case, $C_{gs}$was $3fF$. In less than the area than the otherwise necessary $1pF$ capacitor (28) predicts, the circuit pictured in Figure 26b was laid out.

There are several different potential topologies from which to choose when replacing a floating-gate transistor with a buffer that incorporates stored charge. In the next section, the topology of Figure 26b will be discussed in the context of why it is beneficial for floating-gate programming. For now, we will consider it in terms of the role the output resistance plays on the frequency response and the sizing of the gate capacitor.

The resulting frequency responses for the floating-gate and amplifier cases are shown in Figure 27. The resistance at the gate due to the amplifier defines the worst-case SNR between the source and gate for a fixed set of capacitors and an amplifier biased in deep subthreshold. The worst-case transfer function is given as

$$\left(\frac{Vgate}{Vsoure}\right)_{\substack{worst\\case}} = \frac{sRC_{ov}}{1 + sRC_{ov}}\;;\; R = \frac{R_x}{1 + \frac{C_{fb}}{C_{fb}+C_{fgo}}gmR_x} \tag{31}$$

If the same C_T were applied to the gate of the VMM transistor as in the floating-gate case, the transfer function with the amplifier would have the same maximum gain. However, if the current in the amplifier is biased such that (31) is $20 \cdot log10\,(2^n)\;dB$ down from $-3\,dB$

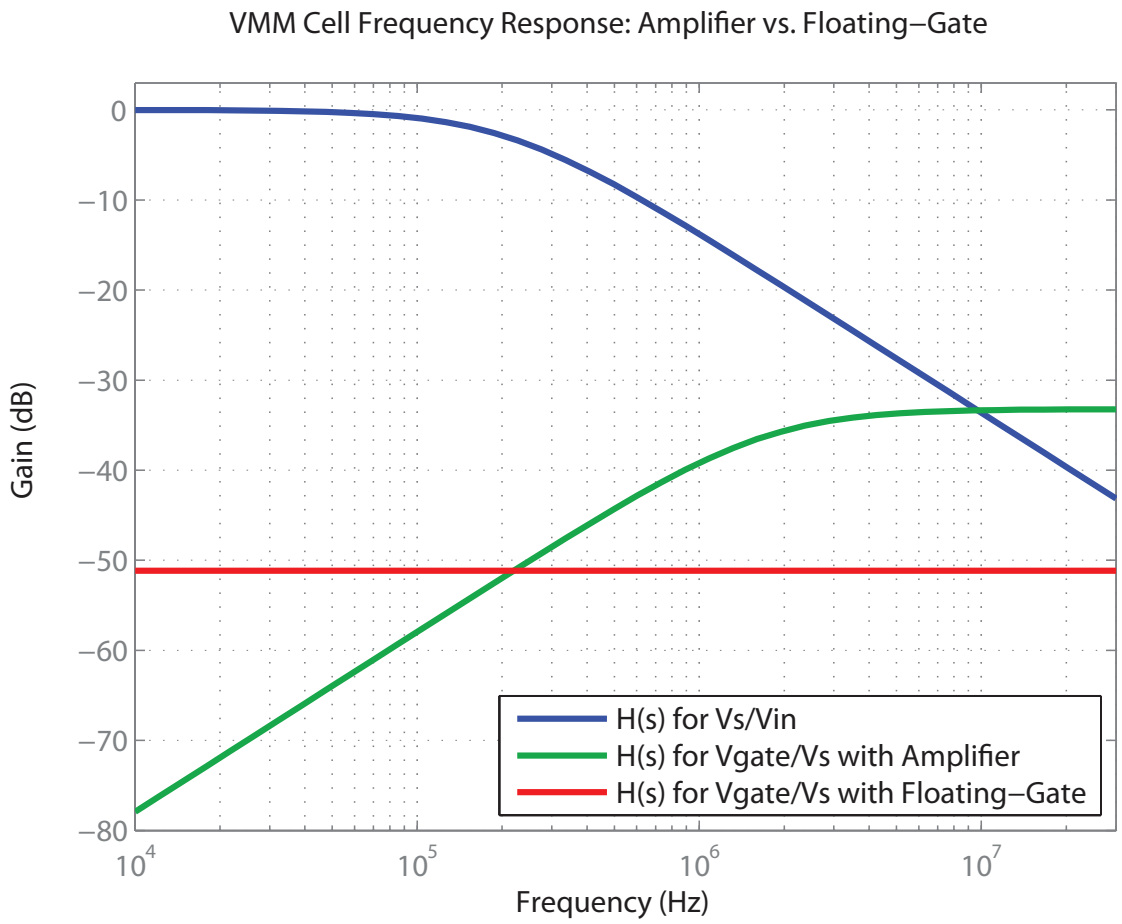


Figure 27. Comparison of frequency response of floating-gate and amplifier computational transistor. For a fixed C_{gs} of $3fF$, the total capacitance on the computational transistor's gate was sized so for an $8\,bit$ accurate multiplication. In the case of the floating gate, $C_T = 1.085pF$. For the amplifier, $C_{T=}102fF$.

at $f_{BWsource}$, then the circuit already has enough precision without relying on the attenuation from an additional $\frac{C_{ov}}{C_T}$. And because the system is strictly decreasing over the range of the source bandwidth, there is virtually no benefit to adding extra capacitance at the gate for the purpose of increasing C_T . The only place where extra capacitance plays a role is in C_{fb}, which relates to how much the gain of the amplifier reduces the output resistance. Under the circumstance that the current required to move the zero in 31 to the desired precision violates the power budget for the circuit, additional gate capacitance is required.

The amplifier allows for trading capacitor area for power in a straightforward way. Because R is roughly inversely proportional to current, the total capacitance and current

can be by traded by the same factor. For the same reason, the power of the amplifier can be doubled for each additional bit of precision. In the case of Figure 27, $8-bit$ precision was achieved at $100nA$ of current and only required about $100fF$ of capacitance on the input node in order to satisfy $f_{BWsource}$.

5.4 SNR

In choosing pico- and nano-ampere current levels for signal representation, the electron density though a transistor and into a sensing structure is small enough that it is necessary to consider the fundamental limit for measuring quanta of charge on the SNR. We can develop intuition about the SNR of a system by considering the signal as the collection of continuous, independent arrivals of charge. Measuring the signal is then a question of counting the number of electrons. This allows us to analyze the count using a Poisson process [34]. If we measure a current I over a sampling period of duration T, we can expect to count the arrival of n electrons, calculated as:

$$n = \frac{I}{q}T \tag{32}$$

The number of expected arrivals provides us the parameter to the Poisson process, $\lambda = n$. In a Poisson distribution, the mean is equal to the variance, which is equal to the distribution parameter λ. So for an SNR defined by μ/σ,

$$SNR = \frac{\mu}{\sigma} = \sqrt{\lambda} = \sqrt{\frac{I}{q}T} \tag{33}$$

We then relate the sample period T to a sample frequency f_s in order to investigate the relationship between signal currents and system speed for different SNR requirements.

$$f_s = \frac{1}{T_s} = \frac{I}{qSNR^2} \tag{34}$$

Equation 34 is evaluated over five decades of current for SNR values from four to nine bits in Figure 28. These ranges are typical for analog subthreshold currents. The Poisson

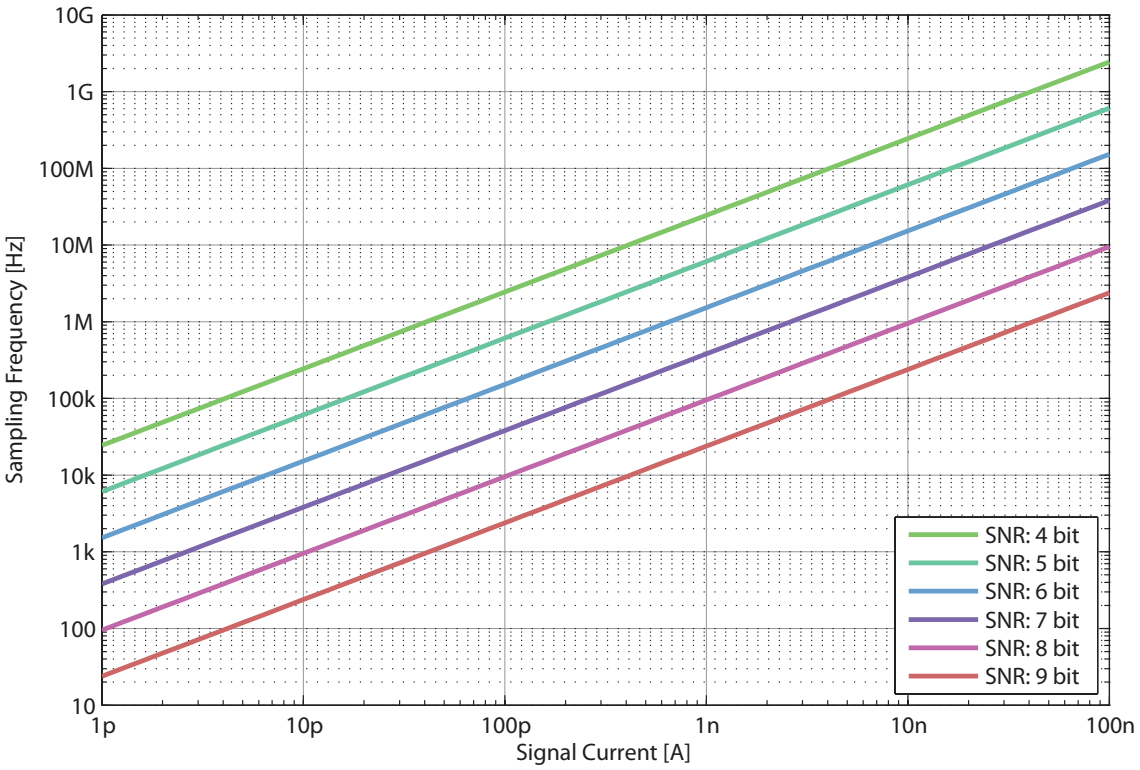


Figure 28. Sampling frequency of current levels with different SNR requirements predicted from Poisson process. This only takes into account thermal noise; the data shown for lower frequences will be have a non-negligible contribution of flicker noise.

process is used here to address thermal noise; the flicker noise will be non-negligible for the lower frequencies.

In order to relate the issue of SNR back to a VMM computational element, we will calculate the SNR of a settled current from the computational transistor flowing into a sensing structure characterized by an input resistance and input capacitance. The portion of the input current flowing into the input resistance is the sensed current, which results in a transfer function from input current to sensed current:

$$I_{sensed} = I_{in}\frac{R||sC}{R} = \frac{I_{in}}{\frac{1}{R} + sC}\,;\ \frac{I_{sensed}}{I_{in}} = H(s) = \frac{1}{1 + s\tau} \tag{35}$$

where $\tau = RC$ of the sensing structure. We then write the sensed noise power as:

$$I^2_{n_{sensed}} = I^2_{n_{FET}}\,|H(j\omega)|^2 = \frac{I^2_{n_{FET}}}{1 + \omega^2\tau^2} \tag{36}$$

where $I^2_{n_{FET}} = 2qI_{signal}\,df$, under the assumption that whatever structure is driving the source and gate of the transistor provides negligible noise. By integrating over all frequencies, we find the total integrated noise power,

$$I^2_{n_{sensed}} = \int_0^\infty \frac{2qI_{signal}}{1 + \omega^2\tau^2}df = \frac{qI_{signal}}{2\tau} \tag{37}$$

We will define the SNR here as the ratio of the signal to the peak noise. Because we are considering a settled input signal, all of the input current will flow into the sensing structure. As a result, the SNR is found to be:

$$SNR = \frac{I_{sensed}}{I_{n_{sensed}}} = \frac{I_{siganal}}{\sqrt{\frac{qI_{signal}}{\tau}}} = \sqrt{\frac{I_{signal}}{q}\tau} \tag{38}$$

Equation 38 is similar to 33. The critical difference to recognize is that τ is not a sampling period. In the thought experiment where we count electrons with an arrival dictated by a Poisson distribution, the sampling period was a free variable. For a real system with a settled input, every measurement at the output of that system will have embedded in it a noise spectrum that was subject to the system bandwidth. In order to increase the SNR,

it is necessary to increase the effective sampling rate by taking additional samples and averaging the result. For instance, if a real system requires 6τ for an output to settle, the measurement taken after 6τ will have an SNR of $\sqrt{\frac{I}{q}\tau}$, not $\sqrt{\frac{I}{q}6\tau}$, because the additional delay did not change the bandwidth constraining the noise. In this case, to increase the SNR by $\sqrt{6}$, six measurements would need to be taken and averaged together.

Section 5.2 discussed the use of a logarithmic amplifer (logamp) for sensing currents at the input and output of a VMM. The τ for a logamp is $\frac{C}{A\,gs}$, where $gs = \frac{I_{signal}}{U_T}$ and A is the open-loop gain of the amplifier. The SNR is then:

$$SNR_{logamp} = \sqrt{\frac{I_{signal}}{q}\frac{C}{A\,gs}} = \sqrt{\frac{U_T C}{qA}} \tag{39}$$

So long as A remains fixed, the SNR will be fixed for all current levels where the flicker noise and other external effects do not dominate. Intuitively, as the signal power increases (or decreases), the noise power will increase (or decrease) linearly with both current and bandwidth. Sensing high dynamic range signals with a logamp is difficult though, because the bandwidth changes with the input signal. In order to fix the bandwidth, for the sake of stability and settling time, the circuit can be modified to adapt the open loop gain A based on the signal level. While this will produce a system with a fixed, or at least improved, settling time for the lower currents, it will come at the cost of reduced SNR at lower signal levels.

Figure 29 is the result of measuring the logamp in Subsection 6.1.4. In Figure 29a, each data point for voltage SNR was the result of taking 1000 measurements at a particular DC current level, and using the log-base-two mean by range of that dataset. The current SNR was computed using the IV relationship of the transimpediance amplifier, and represents the mantissa of a floating-point current measurement. The slope in the SNR is a result of adapting the amplifer gain with input signal level. The effect of gain adaptation is shown in Figure 29b. The settling time for a $6-bit$ resolution measurement was computed for a range of current steps. For an amplifier without gain adaptation, the settling time would have

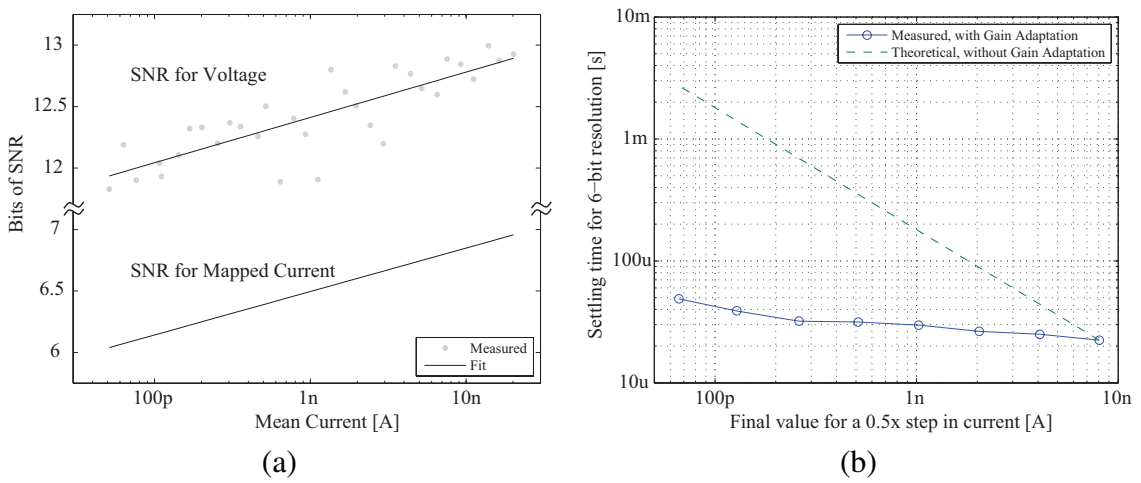


Figure 29. (a) Impact of gain adaptation on SNR of a logamp. Each data point for voltage SNR was the result of taking 1000 measurements at a particular DC current level, and using the log-base-two mean by range of that dataset. The current SNR was computed using the IV relationship of the transimpediance amplifier. (b) Effect of gain adaptation on the settling time. For an amplifer without gain adaptation, the settling time would have doubled for each subsequent current step, instead of increasing by only $\sim$ 10%, resulting in the SNR varation in Figure 29a.

doubled for each subsequent current step, instead of increasing by only $\sim$ 10%, resulting in the SNR varation in Figure 29a.

5.5 Programming

The approach to programming is a critical consideration in constructing a VMM. In both VMM configurations in Figure 21, the computational transistors on the output form an array that is compatible with the discussion from Chapter 3. The way we program those transistors is by altering their DC point until its possible to create the Φ_{dc} necessary for injection. If we continue down the path of Chapter 3, that means the supplies of the IC are ramped up and the drain voltage is pulsed to ground in order to create the necessary V_{ds} for injection. That means for every pulse or array of pulses, the power supply must be moved slowly enough to avoid internal IC transients that could lead to unexpected charge movement on the floating-gates. There is also a substantial complexity penalty; care must be taken to disconnect or disable surrounding circuitry to avoid forward biasing.

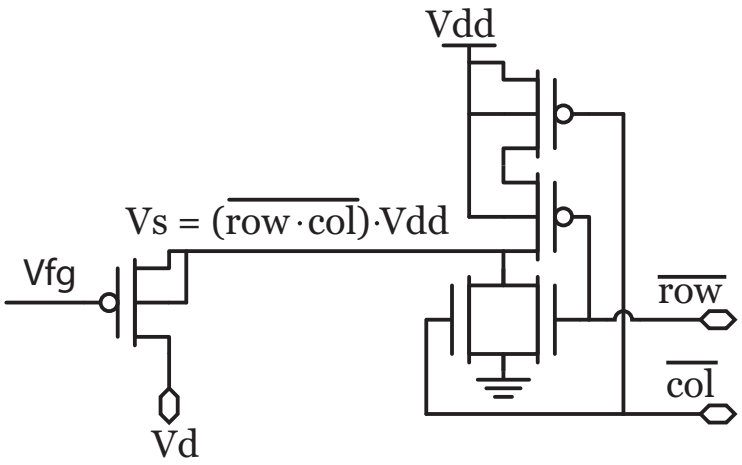


Figure 30. Array selection circuitry for floating-gate with negative drain pulsing.

An alternative scheme trades power supply ramping for pulsing the drain below ground. Negative voltages are applied only to the subset of devices being injected, reducing the burden on the rest of the system. Only the circuitry with the negative potential is disturbed– by using multiple negative potentials different portions of an IC can be programmed without interrupting the system function. By limiting the negative voltage to only the device being injected, the supply generating the negative voltage needs to support only the current being injected. This is critical for when the source is an on-chip charge pump.

Because these devices are used in an array, it is still necessary to provide some means of individual selection. Rather than use the intersection of a sufficient gate-to-source potential for a channel and a sufficient drain to channel potential for a high field, which doesn't provide proper isolation, we can use the circuit in Figure 30.

In Figure 30, an analog NOR gate and a floating-gate pFET are pictured. The output of the NOR gate controls the source of the floating-gate transistor. The premise of the structure is that by only by asserting both active-low signals row_bar and col_bar, does the source get a potential other than ground. In an array structure, that means that by only asserting a single row and column, only a single device will have a non-zero source voltage.

The insight of the effort in Section 3.2.2 was that in order to prevent parasitic charge

movement, the fields at the active regions of the transistor must be kept low enough to prevent hot-carriers and FN-tunneling. The choice of where to connect the bulk potential was critical for preventing parasitic charge movement. If the bulk had been left at the supply, parasitic charge movement would have occurred in the high field between the drain and the substrate. By connecting the bulk to the source, only the transistor selected for injection will be exposed to a high-field. The GIDL current on non-selected transistors resulting from the large gate-to-drain difference during injection is minimized because of the low bulk potential of the unselected device. As a result, the drain is never pulsed lower than $-2.5V$; $2.5V$ is characterized as a voltage drop safe for $.35um$ CMOS. The only potential source of charge movement is FN tunneling across C_{GD}, which is a function of where the floating-gate voltage is kept.

A fundamental difficulty with programming a floating-gate transistor with CHE injection is that as carriers inject into the gate, the effective floating-gate voltage drops. That drop corresponds with an increase in current through the channel, which results in more carriers for injection. That situation is described analytically by the following:

$$I_{inj} \propto I_{channel} = I_b e^{-\kappa \frac{V_{fg}}{U_T}} \tag{40}$$

where the device is operating in subthreshold. As the channel current increases, the device will self-limit the injection current through the potential difference that forms in the channel as the device moves above threshold, but it still requires orders of magnitude change in current in order to reach that self limit, shown experimentally in Figure 8. One way to handle the explosion in injection current over time is to use the modeling technique of Chapter 2. In that case, $\frac{\Delta I}{I}$ is related to a pulse length and drain voltage. The high-order system that results provides a means for hitting precise targets, but only through a gradual, multi-step approach. Keep in mind that to reprogram a floating-gate device, it along with every other device sharing that global tunneling line must be modified. As a result, the effective threshold of every transistor will be need to be set to the worst-case high threshold required in the system, meaning that the transistors requiring the worst-case low threshold

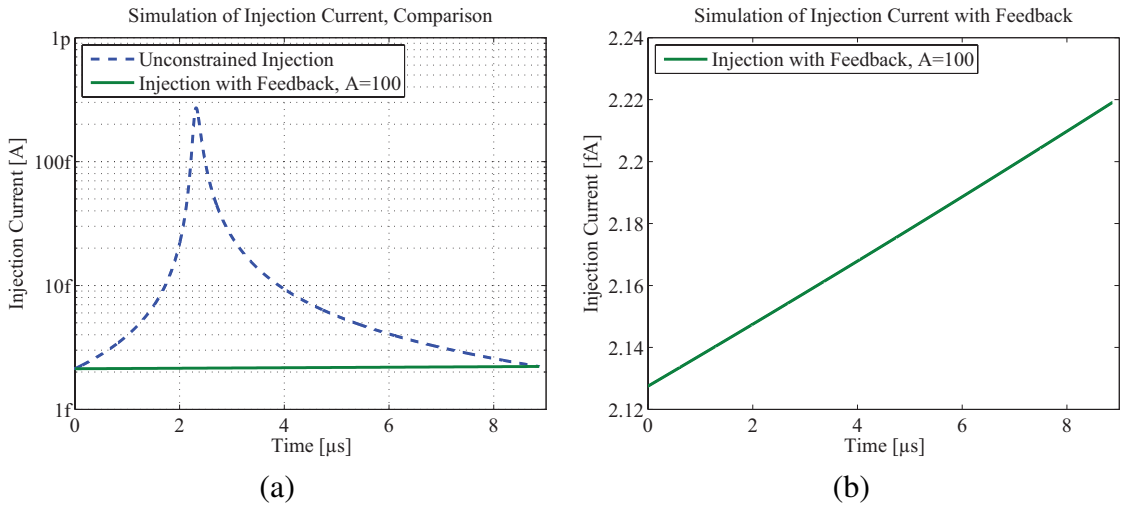


Figure 31. Impact of feedback on the injection current. The simplified EKV expression is used for modeling the channel current and the model from Chapter 4 is used for modeling the injection current.

will require a substantially different floating-gate voltage. For instance, if two transistors sharing the same tunneling line require a $1V$ difference in threshold, they will require a $1V$ difference in floating-gate voltage. Considering the injection efficiency issues, the gate voltage will have to be changed several different times to move the bias point back to a place where injection can take place. And once the device is biased at a point where it can be injected within the mapped region of operation, there is still a limit to a safe step in charge because any error in the measurement the starting current will be subject to the exponential of (40).

The fundamental limitation in programming over a wide range of effective floating-gate voltages is then the limitation of maintaining a consistent injection current. The insight to then take from (40) is that by attenuating the change in the floating-gate voltage, we can attenuate the variation in the injection current over time. Figure 31 is a simulation of the injection current for a floating-gate transistor using the device parameters of Chapter 4. Figure 31a shows the comparison of a transistor injecting without any explicit feedback and a transistor with an ideal amplifier attenuating the change in the floating-gate voltage by $40\,dB$.

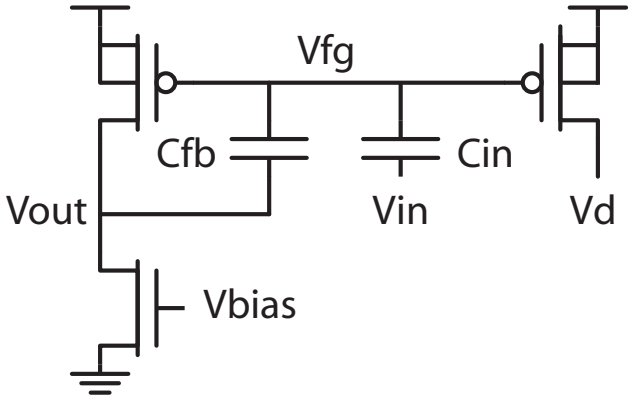


Figure 32. Feedback circuit for fixing V_{fg} during injection. The transistor on the right is the injection transistor.

The circuit in Figure 32 represents a compact method for fixing V_{fg} [35, 36, 37]. There are two ways of looking at this circuit: it is a current mirror with I_{bias}, due to V_{bias}on the nFET, flowing through the both transistors; or it is an amplifier with the feedback capacitor C_{fb}. Consider first the circuit as a mirror. The current through the injection transistor is fixed, which means V_{fg} is fixed. I_{inj} is a function of the channel current and the floating-gate potential—with both values fixed, the injection current is fixed as well. As an amplifier, the circuit represents a way of attenuating the change in the floating-gate voltage by the gain.

The capacitors in Figure 32 must be chosen with respect to the programming and isolation requirements. With respect to programming, the impact of a change in charge can be found by solving KCL at Vout.

$$Ibias = I_0 \cdot \mathrm{e}^{\frac{\kappa(Vdd-Vfg-|Vt|)}{Ut} + \frac{Vdd-Vout}{Va}} \tag{41}$$

where Vfg is given as

$$V_{fg} = \frac{C_p}{C_t} V_p + \frac{C_{fb}}{C_t} \cdot Vout + \frac{Q}{C_t} \tag{42}$$

and C_t is the total capacitance at the floating-gate node, C_p represents the total parasitic capacitance at the floating-gate node through which an effective voltage V_p couples, and Q is the stored charge at the node. Solving for V_{out} and then differentiating with respect to the

charge,

$$\frac{dV_{out}}{dQ} = -\frac{\frac{\kappa V_a}{C_t U_t}}{\frac{\kappa V_a}{C_t U_t} C_{fb} + 1} \approx -\frac{1}{C_{fb}} \tag{43}$$

By choosing the smallest number of electrons to move at a time during a programming pulse, Q_{min}, and the minimum desired resolvable voltage change on V_{out}, ΔV_{out_min}, the minimum value of C_{fb} must be $\frac{Q_{min}}{\Delta Vou\_min}$. The value for C_{in} is then determined by the maximum voltage swing necessary at the output from a change in the control voltage V_{in}.

$$\frac{dV_{out}}{dV_{in}} \approx -\frac{C_{in}}{C_{fb}} \tag{44}$$

If the output voltage is being used to set the gate voltage for a computational transistor in an array like Figure 23, the output must be able to swing between the programmed voltage for the largest current output and the voltage to reduce the transistor current output to leakage.

5.6 Implementation

An ideal floating-gate computational cell has no programming time, consumes no area, requires no external instrumentation, and does not adversely impact the circuit it is associated with. When implementing a floating-gate memory element, each of the aforementioned ideals are relaxed in order to implement a real circuit. However, we can approach the ideal with well-conceived design procedures. The circuit in Figure 33 represents a desirable implementation of such an element. It is an integration of the issues discussed in the previous sections of the chapter.

The first critical design decision for this circuit is use of a negative voltage for implementing the Φ_{dc} necessary for injection. In many of our previous implementations, the supplies of the IC are ramped up and the drain voltage is pulsed to ground in order to create the necessary V_{ds} for injection, but that comes at the cost of speed and complexity. Next, an analog NOR gate is used to address the device to be injected. By using a comparator to measure the output of the computational cell during the negative pulse phase, the system output can be directly programmed and gated. One other issue with floating-gate use in the

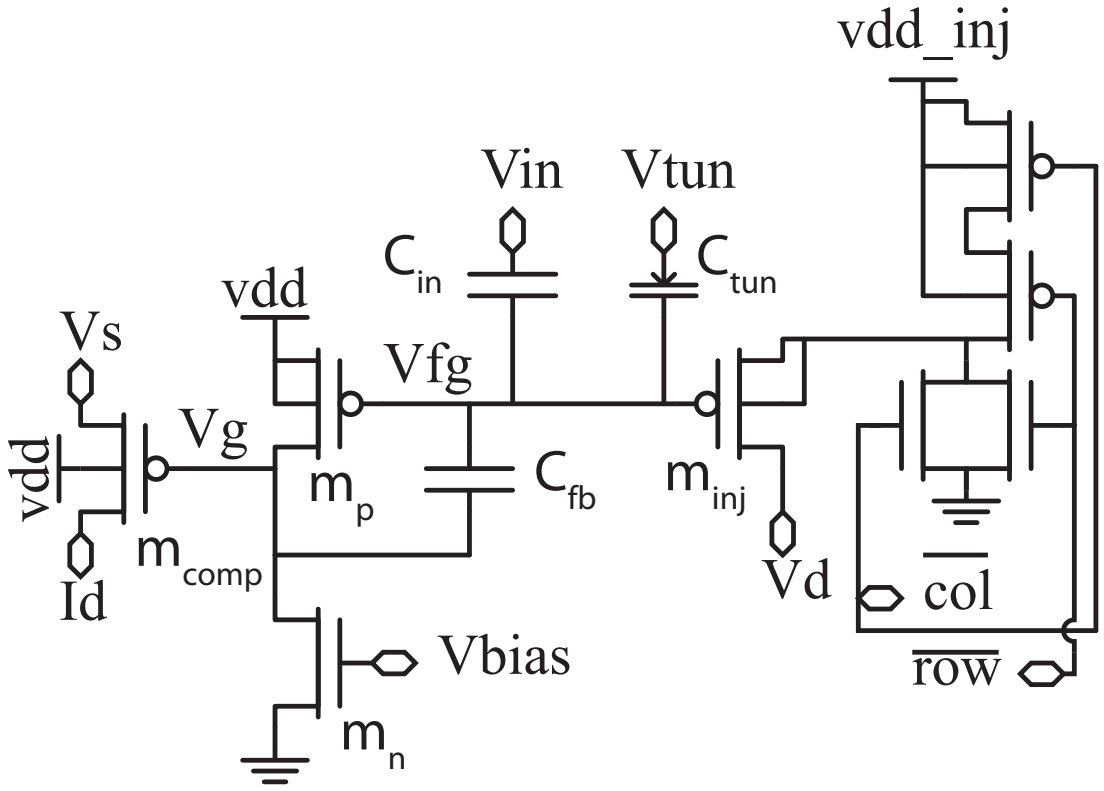


Figure 33. A complete floating-gate cell. An amplifier formed from Mp and Mn is used to drive the gate voltage of the computation FET, Mcomp. The amplifer also serves to fix the voltage of the floating node, Vfg. Injection is performed by applying a negative voltage to the drain of Minj and a positive voltage to its source and bulk. The source and bulk are controlled by an analog NOR gate.

signal path is that potentially undesirable coupling occurs through the source and drain of the fg-pFET. We can alleviate this issue by using the amplifier formed by Mp and Mn as an internal node that applies the programmed voltage to Mcomp. Precision programming is key to most analog floating-gate use. We can simplify our programming methodology by stabilizing the injection current through the use of a diode connection on Mp and a current source, Mn. As current flows onto the floating-gate through Minj, the capacitor feeds back the voltage necessary to maintain the current. As a result, the floating-gate voltage and source current of the device being programming is fixed and so is the injection current.

In order to facilitate continuous-time injection, a logamp was used as the readout circuit for the computational transistor. In addition, the circuit pictured in Figure 34 was used. Targeting a particular current was just a matter of latching a stop condition, sampling a reference voltage that corresponded to a particular current level, and the allowing the transistor to inject. The output of the IV converter increases or decreases based on the sign of the current input, which is why the XOR gate and D-latch were necessary. Upon enable, the row select signal is asserted. When the injection is taking place, after enable has been de-asserted, the row signal is only de-asserted when the output of the comparator is not equal to its value stored during the period enable is asserted.

5.6.1 Simulation

In order to examine the transient response of the circuit during programming, we can the use the model described in Chapter 4. We simulated the schematic from Figure 33 with a vcvs that de-asserted the row signal when Vout moved past 1.5 V. The transient simulation was started with 1V on the floating-gate node. The Vd terminal was moved from the DC potential of ground to three different negative potentials, and left negative for the duration of the simulation. The transient response of the simulation is shown in Figure 35. During the first micro-second, the injection current remained off until Line 1 of Figure 17 becomes true. At that point, injection current is applied to the gate. As expected, the injection current decreases for higher Vd potentials. When the vcvs trips, moving source/bulk potential

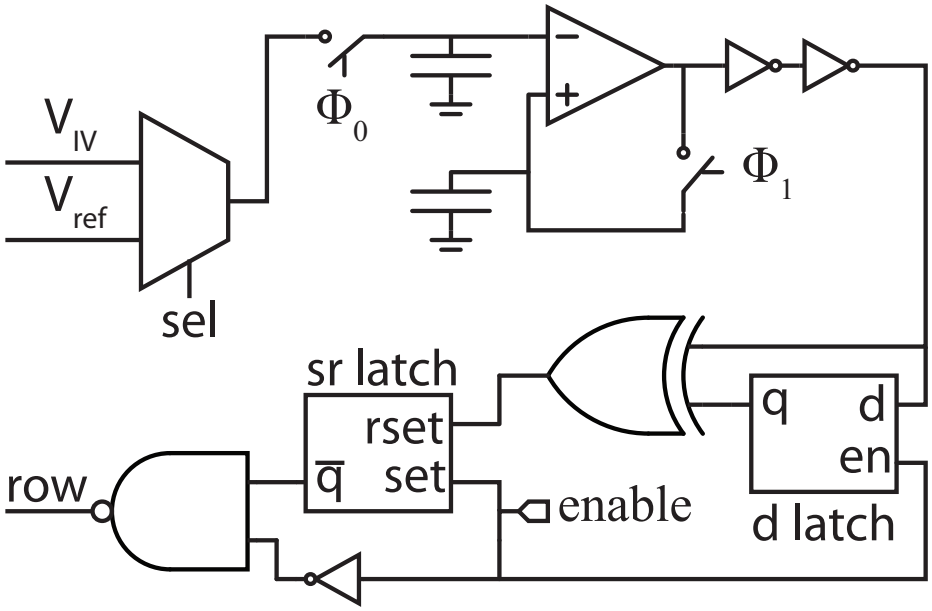


Figure 34. Comparator circuit with row logic for continuous-time injection.

down, the injection current drops back to zero. The source current is held relatively fixed throughout the injection phase, as expected. In addition, the output potential (Vg) increases linearly to the target potential. Because the bulk potential is brought to ground in order to stop injection, there is a significant step in the Vg. As a result, precision use would require a small amount of characterization in order to account for the bulk coupling.

Post fabrication, our goal was to experimentally determine the injection current, I_{inj}, and the bias-dependent injection term, V_{inj}, in order to characterize the programming and back-annotate the simulation for future development. In the experiment, we have a measurement for the output of our IV converter changing over time, Figure 36a. In order to convert the output voltage to a current though the computation FET, we apply a known test current to the VMM summation line. There is no benefit to using the analytical solution, since we have the actual I-V characteristic, discussed in Chapter 6. The IV relationship gives us a VMM current, Figure 36b, which we relate back to an voltage output of the floating-gate cell though the subthreshold expression for the transistor. By rearranging, we

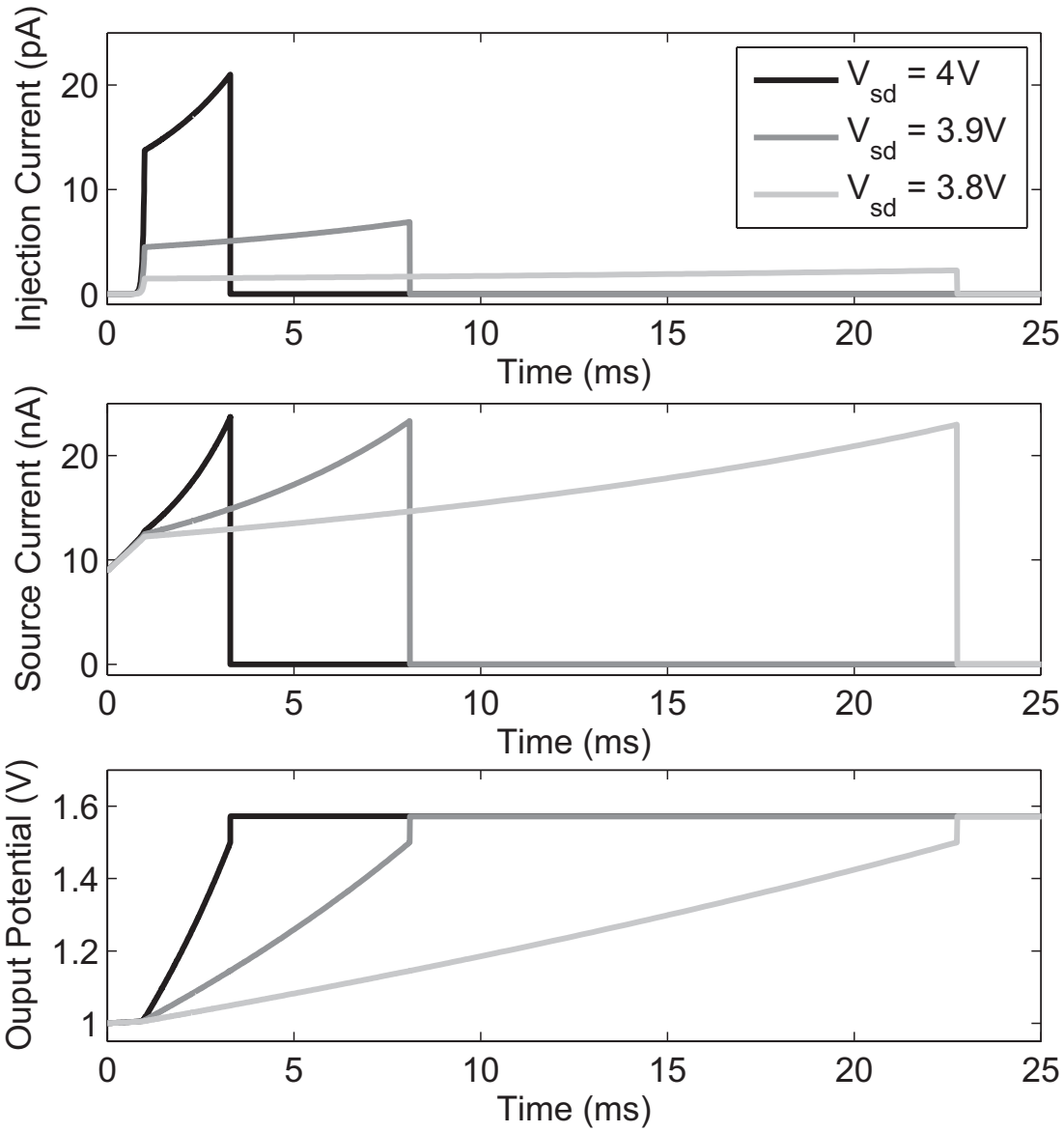
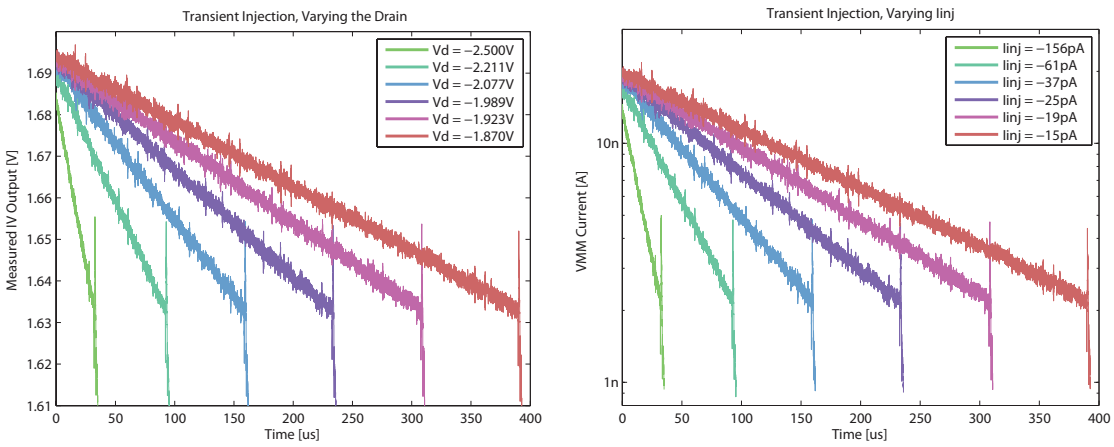


Figure 35. Results of simulating the circuit in Figure 33. The output of the high-gain amplifier and gate current and source current of Minj are shown for a drain potentials -1.5, -1.4, and -1.3 volts.



(a) (b)

Figure 36. Measured results of fixed current injection. (a) Voltage measurement (from IV Converter) of continuously injecting VMM cell for different drain voltages. The large spikes in the data are a result of the on-chip comparator tripping, signaling an end to the injecting and de-asserting the voltage causing the high-field. (b) Post-measurement mapping of output voltages to VMM cell currents. The rate of current change was mapped to injection current through (47)**.**

find that

$$V_{out} = -\frac{U_t}{\kappa} \ln\left(\frac{I_{out}}{I_b}\right) \tag{45}$$

where I_b is a collection of all the terms of transistor not related to the applied gate voltage. In order to determine I_{inj}, we will use the fact that $\frac{dV_{out}}{dt} = \frac{dV_{out}}{dQ} \cdot \frac{dQ}{dt}$. Using our result from (43), recognizing that $^{dQ}/_{dt}$ is the injection current, and by taking the derivative of (45) we can write

$$\frac{dV_{out}}{dt} = -\frac{U_t}{\kappa} \frac{\frac{d}{dt} I_{out}(t)}{I_{out}(t)} = \frac{dV_{out}}{dQ} \cdot \frac{dQ}{dt} \approx -\frac{1}{C_{fb}} \cdot I_{inj} \tag{46}$$

We will approximate the derivative using our measured data, yielding

$$I_{inj} \approx \frac{C_{fb} U_t}{\kappa \Delta t} \cdot \frac{\Delta I_{out}}{I_{out}} \tag{47}$$

Our calculation for I_{inj} is limited by our estimation of κ, which varies with current, U_t, which varies with temperature, $\frac{dV_{out}}{dQ}$, which depends on the gain and estimation of C_{fb}, and the time step, which must be small enough approximate the rate of I_{out} changing.

The results from Figure 36 allow for determining V_{inj}. Each V_d is associated with a particular I_{inj}. In addition the relationship between the drain voltage and the injection

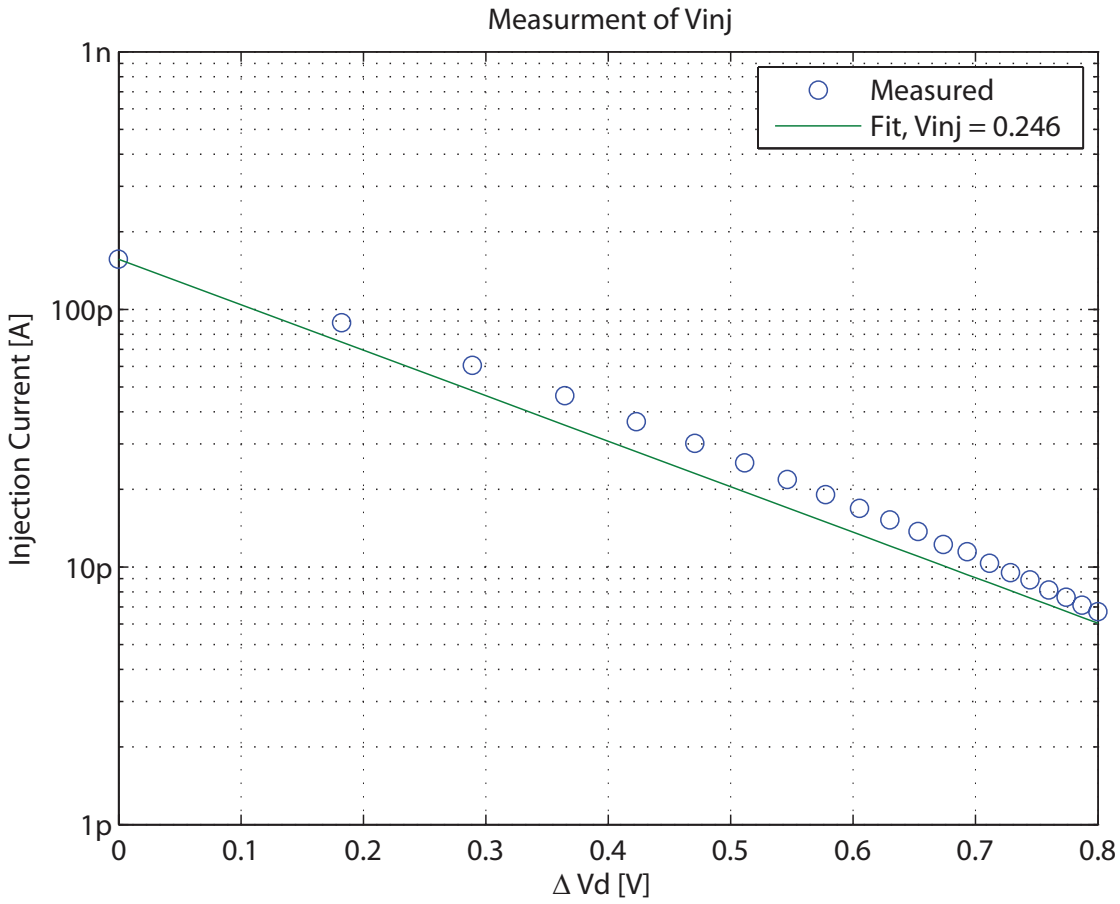


Figure 37. Experimental measurement of V_{inj}.

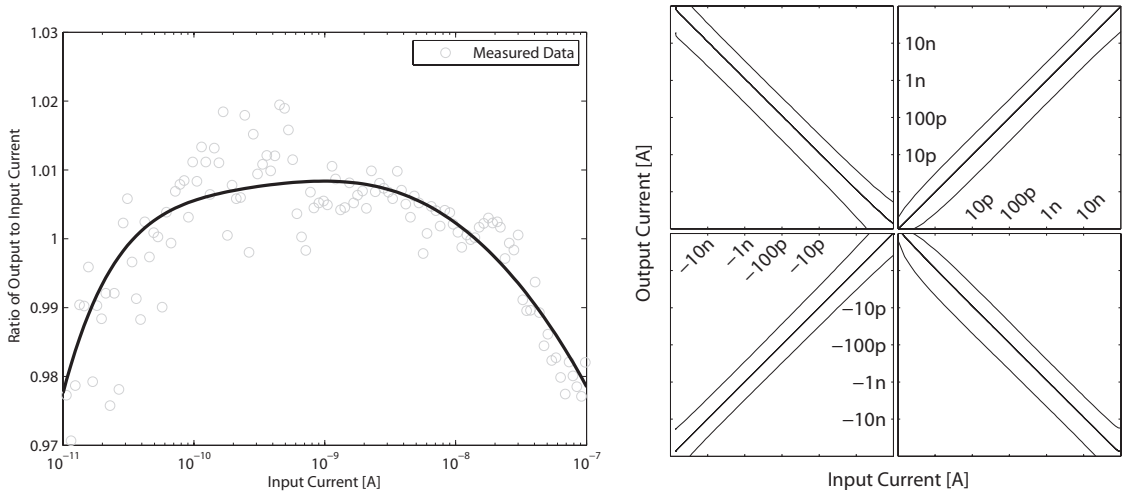
current is:

$$\frac{dI_{inj}}{dV_d} = \frac{I_{inj}}{V_{inj}} \tag{48}$$

The linear fit value for V_{inj} is shown in Figure 37.

5.6.2 Experimental Results

We have measured and characterized the VMM of our computational transform imager, discussed in Chapter 6. The converter we employed for current measurement was the bidirectional logamp discussed in Subsection 6.1.4. In order to evaluate the input-output linearity, a single VMM cell was programmed to unity current gain. The ratio of the output current to the input current was plotted against input current, as shown in Figure 38a. Over the center 2.5 decades of plotted current, the error is less than 1%, and the entire four



(a) (b)

Figure 38. Measurement of VMM multiplication, single-ended and four-quadrant. (a) Single-ended measurement of a multiplication using the VMM cell programmed to a 1x multiplication. The output current was divided by the input current, resulting in the gray circles. A fit of the measured data was provided in order to better visualize the trend over the four orders of magnitude. Over the center 2.5 decades of plotted current, the error is less than 1%, and the entire four decades fall within $\pm 2\%$ error. The high-end accuracy degradation was due to the test transistor leaving subthreshold, while the low end was thought to be limited by the input current leakage–the test current was applied through a $256x256$ pixel plane. (b) Measured data showing four-quadrant multiplication using two input transistors and four VMM cells programmed to the differential weights:$[-1/6, 1/6, -1, 1, -6, 6]$.

decades fall within $\pm 2\%$ error. The high-end accuracy degradation was due to the test transistor leaving subthreshold, while the low end was thought to be limited by the input current leakage.

In order to implement a full, four-quadrant multiplication, VMM cells were grouped into sets of four. Consider Figure 23b. By setting the weights of transistors $m_{i,j}$ through $m_{i_{i+1,j+1}}$ to $\begin{bmatrix} 1+w/2 & 1-w/2 \\ 1-w/2 & 1+w/2 \end{bmatrix}$, and establishing the convention that I_i and I_{i+1} are positive and negative portions of a bidirectional input current and I_j and I_{j+1} are positive and negative portions of a bidirectional output current, a fully differential VMM is implemented, where w is the weight of the quadruplet. We programmed such a quadruplet to show gain and attenuation for positive and negative weights. The results are shown in Figure 38b. Note that rather than use the linear "X" plots that are typical of multipliers, we have instead

plotted our input-output measurements on a logarithmic scale in order to capture the wide dynamic range capability of the multiplication. The VMM is discussed in more detail in the context of our imager in Subsection 6.1.3.

By using a combination of the techniques discussed in this and previous chapters, we were able to program the VMM weights to within the limit of our measurement circuitry in a respectably short period of time, a breakdown of the time will follow. With respect to the measurement circuitry, we had a $400\mu V$ noise floor at the output of the bidirectional IV. If we consider the accuracy of our measurement in terms of the log-base-two ratio of the mean to the range of 1000 measurements, the 12 to 13 bits of accuracy we see in voltage translate to 6 to 7 bits in current over more than 2.5 decades of current. It is useful to think of the SNR in current as the mantissa of a floating-point measurement, with another 6 to 7 bits for the exponent.

The actual programming of the VMM cell array was broken up into three phases: *bring into range*, *coarse inject*, and *fine inject*. In order to simplify the programming scheme, a fixed voltage was applied to all of the sources of the computational FETs in the VMM during programming and weight measurement. This was functionally equivalent to setting the same input current for all of the computational cells. It is valid because floating-gate transistors were used in the source buffers for equalizing the offset across all of the rows. We used the offset programming technique described in [38].

For an array of floating-gate transistors where the previous charge state is unknown, the *bring into range* step involves a global erase of a collection of VMM cells and a subsequent injection. Because the VMM cell uses an inverting amplifier to drive the computation transistor, a tunneled cell results in large current flowing through all of the output FETs. The initial injection moves the charge to the point that the control gate can be used to cut off the output current. The result of bringing the devices into range is illustrated in Figure 39. A single, fixed voltage was targeted using the comparator previously discussed in order to optimize for speed. Because a fixed reference voltage was targeted, offsets between the IVs

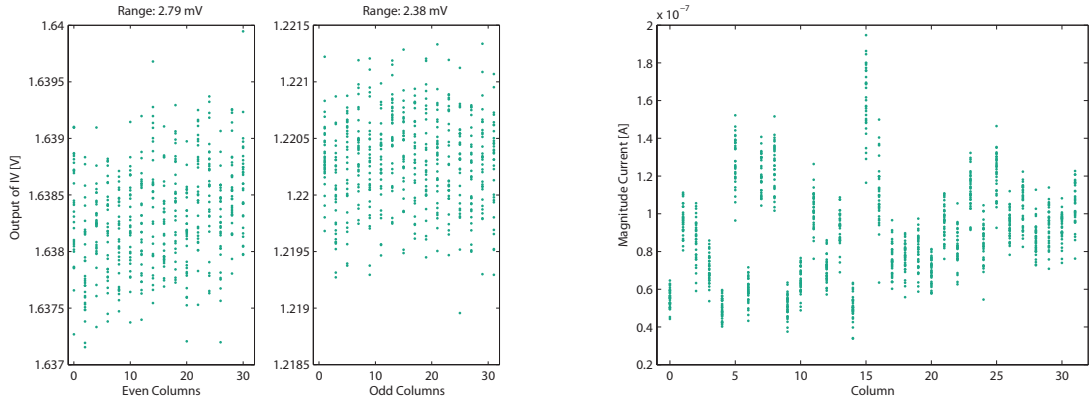


Figure 39. Result of the *bring into range* step in both output voltage and current. (a) After tunneling, the transistors are injected with a fixed reference target for the comparator at the maximum rate of injection, $\sim 500pA$, resulting the measured data pictured. Different targets were necessary for the positive and negative currents. (b) Magnitude output current from all 1024 VMM cells. Because a fixed reference voltage was targeted, offsets between the IVs of each column result in offsets in the current programmed.

of each column result in offsets in the current programmed. For a single array, 1.2 seconds were consumed for the total *bring into range* process. $300ms$ were spent on the high-field exposure for FN tunneling, and the rest of the time was consumed with programming at $\sim 500pA$ injection currents in a repeated process until all the devices were individually addressable. If the previous charge state of the VMM cells is within the range where tunneling does not destroy isolation, the injection step is not necessary, as the devices will already be in range after the tunneling. In that case the high-field exposure is reduced to $70ms$. With the overhead from Matlab instrumentation to an embedded soft-core processor controlling the PCB components, the total time is $140 - 160$ ms.

The *coarse inject* step was used for bringing the device to within 50% of the final current. It involved setting setting a particular target voltage for the comparator associated with the desired target current and allowing the VMM cell to inject continuously until reaching the target. Measured transient results for a device programmed to $250pA$ and $20nA$ after being brought into range are shown in Figure 40. The circuit in Figure 34 is used to select a

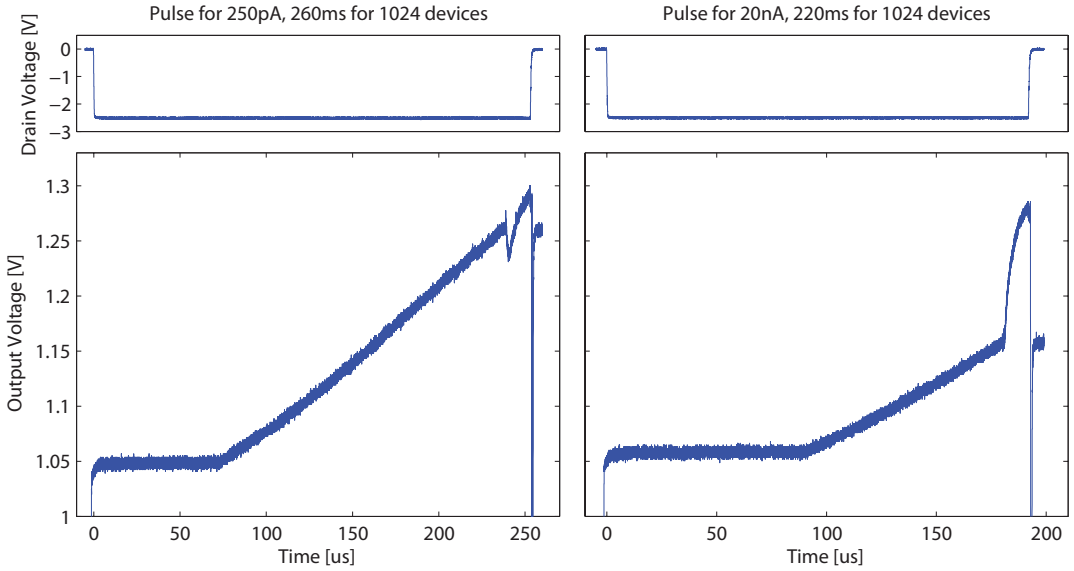


Figure 40. Characteristic measured transient response of a device programmed to $250pA$ **and** $20nA$**, left and right respectively. The output voltage appears fixed initially as the high-current saturates the IV. The spike in the transient output corresponds to the comparator tripping and causing the transistor to become de-selected. The programming time varies from device to device, with a mean time of** $260ms$ **to program all 1024 devices to** $250pA$ **and** $220ms$ **to program the devices to** $20nA$**.**

device, then the drain of the injection transistor is dropped to begin programming. The output voltage remains fixed for the duration that the current from the VMM cell saturates the IV converter. The device continues injecting until the comparator trips and the VMM cell is disconnected. The output voltage of Figure 40 spikes due to there not being any current feeding the IV converter. Once the instrumentation hardware recognizes that the comparator has tripped, the drain is brought back to a point that disallows injection and the circuit is reset for the next device. The temporal results in the plot are characteristic; the programming time varies based upon the starting point, influenced by the tunneling procedure, and the specific device characteristics. The worst case measured time for programming all 1024 devices of the VMM to $20nA$ and $250pA$ were $220ms$ and $260ms$ respectively. The time is the total system time, including external instrumentation.

One of the key issues with the continuous injection is that the drain voltage on the injection FET must be set to a different voltage during programming than during computation,

otherwise there would be a static current flowing through the drain of the injection FET. The drain voltage couples through the gate-to-drain overlap capacitor, resulting in an offset in the output voltage, since the gate is being held fixed. For an output voltage driving a computational FET, which is the condition in this system, the offset results in a current multiplication. A histogram of the resulting multiplication is shown in Figure 41a. For the set of 1024 VMM cells, the variation is about 10% around a factor of about 100. The multiplication is corrected for in the first order by shifting the fixed source to implement a factor of 100 division. The individual offsets between devices are corrected for by performing a linear fit per device. Such a fit takes the form of Figure 41b. The experiment performed was as follows: set a target voltage for the comparator, allow the device to inject until the comparator trips and disables injection, measure the current with the bias point for computation, repeat for the next target. The plot shows the measured voltage and the calculated current from the output of the IV converter.

Next, the 1024 VMM cells were programmed to currents logarithmically spaced from $150pA$ to $20nA$ using the coarse injection scheme, Figure 42a. The external instrumentation overhead consumed $150us$ per device, and the actual injection took between $60us$ and $150us$, for between $20nA$ and $50pA$, respectively. The error from the coarse injection is plotted in Figure 42b. The curvature for low currents is a result of the IV converter bandwidth limit, and can actually be characterized by using a higher-order fit to reduce programming error, but the goal was to maintain a first-order fit because it is more efficient for an embedded implementation–in this case a matrix multiply and a matrix addition to calculate the applied voltage from the voltage associated with a particular measured current.

The coarse injection method can be used to directly program weights where 5-10% error is acceptable. Figure 42c is an image of the measured programmed 4-quadrant HAAR transform, useful for the imager discussed in the next chapter. Because the transform can be defined with the differential weights [1,-1,0], the coarse programming is sufficient to

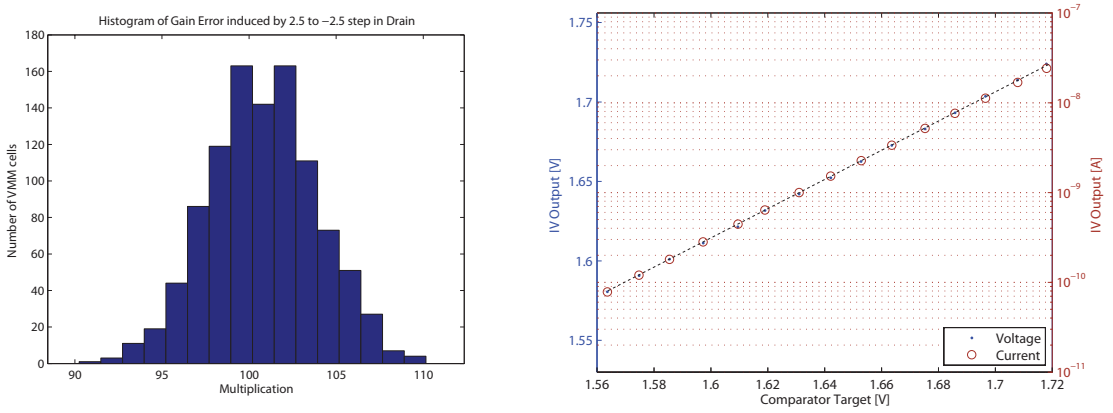


Figure 41. Comparator-based continuous-time coarse programming characterization. (a) Histogram of effective multiplication resulting from stepping the injection-FET drain from the run time potential to the programming potential. Continuous injection means applying a bias point that is different than the run time bias point. In particular, the drain voltage of the injection FET is at a negative potential rather than the positive supply. For an output voltage driving a computational FET, that correlates to a multiplication. Over 1024 devices, the devices vary by about 10% around a factor of about 100. The multiplication is corrected for in the first order by shifting the fixed source to implement a factor of 100 division. The offset between devices is corrected for by performing a linear fit per device. (b) Mapping between comparator targets and measured output voltages and currents for a single VMM cell. The dashed black line represents the linear fit performed to provide the link between the run time bias point and the programming bias point.

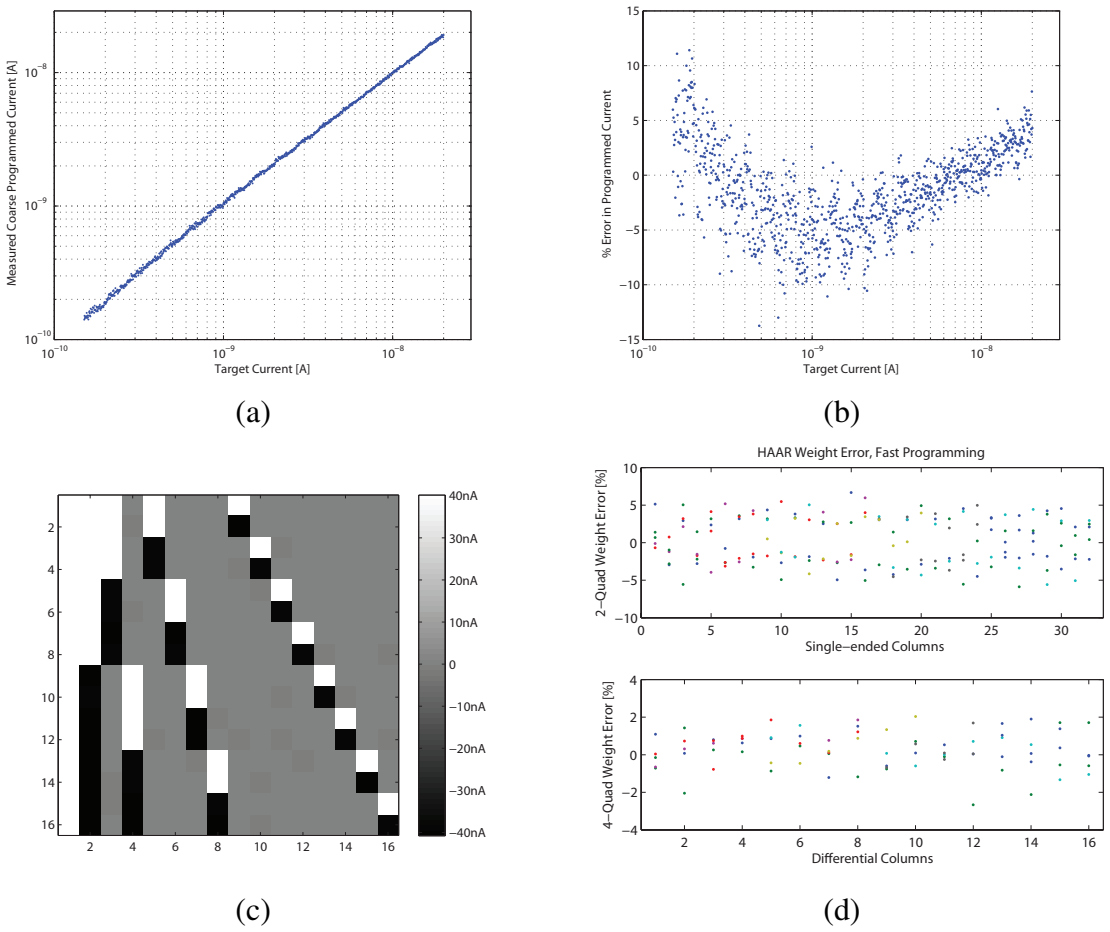


Figure 42. (a) Result of programming 1024 VMM cells to currents logarithmically spaced from $150pA$ to $20nA$. (b) Error associated with coarse programming using a linear fit as a predictor of the target current. The curvature for low currents is a result of the IV converter bandwidth limit. A higher-order fit could be used to reduce programming error. (c) Measured programmed 4-quadrant HAAR transform using coarse programming. (d) Error associated with the programming of 42c. The zero weights, while not shown, were programmed to greater than three orders of magnitude lower than the non-zero weights.

achieve greater than five bits of precision, shown in Figure 42d. Depending on the application, the error in programming can be measured and utilized to eliminate the need to spend programming time.

The *fine inject* step was used for bringing the VMM weights to their final value. This was accomplished through a measure and inject approach. The characterization for the fine programming is shown in Figure 43. A device was initialized to the maximum current of interest, and then the drain of the injection FET was pulsed. After measuring the resulting current, the drain was pulsed again until the operating range of the device was exhausted. This experiment was repeated for different pulse lengths in order to demonstrate that the rate of change of current was independent of the pulse time. The characteristic rate for each device was measured, and the rate array was used in conjunction with the difference of the target and current programmed values of the VMM cells in order to determine an array of pulses. Because all of the devices are brought to within 50% of the final value by the coarse injection, the worst-case field exposure time using the fine programming scheme for the array is bounded by 1024 devices times the field exposure time for a single device that is 50% from its final value. For a single device, the worst field exposure for the biasing in Figure 43a is $66us$.

In Figure 43b, the devices in the VMM were programmed to show a sine wave in log of the currents as an example of fine programming. Bring into range and coarse programming were employed before applying fine programming, which required less than three or less pulses. The error from programming is plotted in Figure 43c. Experimental results of four-quadrant programming of the VMM array to the coefficients of a discrete cosine transform are pictured in Figure 43d. The SNR of the programming was 6.1 bits, limited by the 400 uV noise floor measured at the output of the IV converters, Figure 29a.

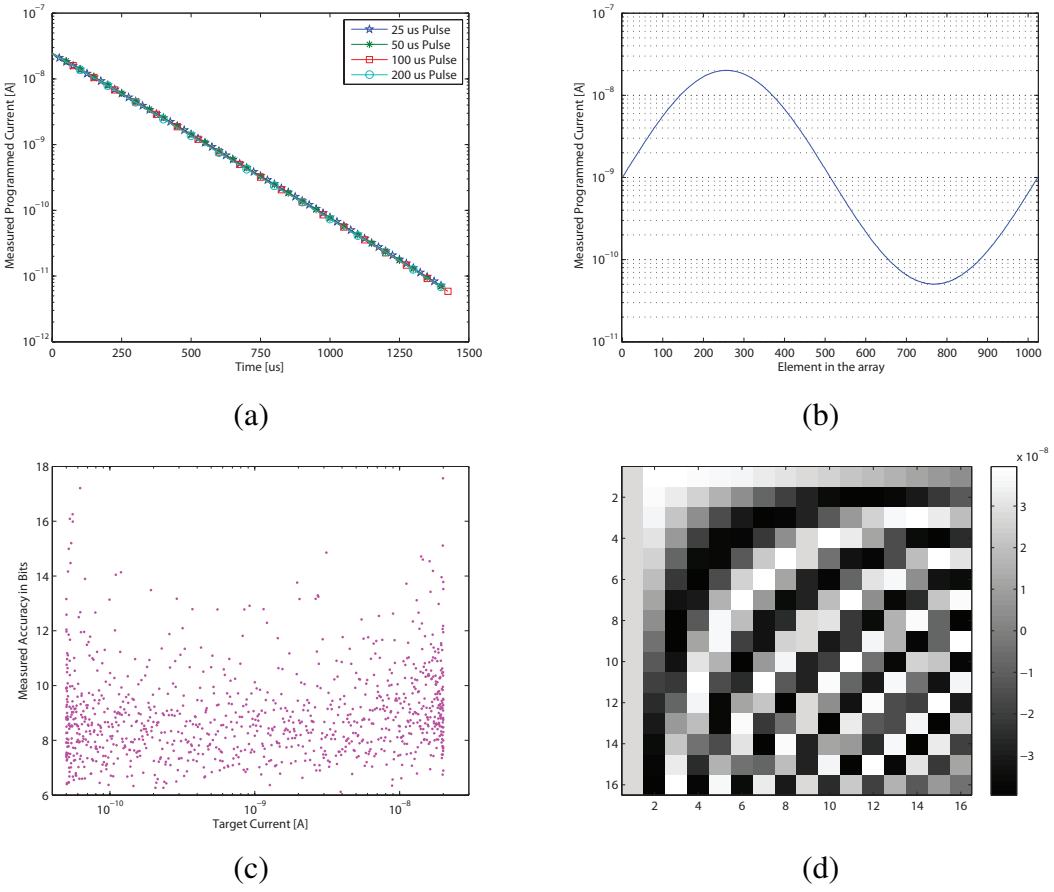


Figure 43. Fine programming characterization and measurements. (a) Measured programmed current for different pulse durations. The key is that all the applied pulse lenghts provide the same rate of change, which demonstrates that the injection current is fixed. (b) The devices in the VMM were programmed to show a sine wave in log of the currents as an example of fine programming. Bring into range and coarse programming were employed before applying fine programming, which required less than four pulses. (c) Measured accuracy in bits of the single-ended programmed currents of the sine wave, which was strictly better than 6 bits. (d) Experimental results of four-quadrant programming of the VMM array to the coefficients of a discrete cosine transform. The SNR of the programming was 6.1 bits, limited by the 400 uV noise floor measured at the output of the IV converters.

CHAPTER 6

REPROGRAMMABLE ANALOG SYSTEMS

By using the techniques discussed in the previous chapters, we can construct large-scale analog systems. In this chapter, the focus of the discussion is on reprogrammable designs. I draw a key distinction between reprogrammable systems and reconfigurable systems. In a reprogrammable system, the circuit architecture is fixed, but the functions and characteristics of that architecture are malleable. For instance, the computational image sensor of Section 6.1 implements two block transforms, one in each dimension of the image being sensed by the pixel plane. The degree of freedom provided by the floating-gate technology is in the ability to set arbitrary transforms–the computational architecture itself is fixed. The adaptive filter in Section 6.2 represents a similar situation, where the filter topology is fixed, but the floating-gate synapse weights are reprogrammable. We build reprogrammable systems to bound system complexity and focus the design exploration. In the following chapter, I will address fully reconfigurable systems, where floating-gate transistors provide a means for redefining the effective system architecture.

6.1 Transform Imager

Sensing and processing are typically well isolated approaches, with an analog transducer on the sensing end and a DSP on the processing end, an ADC sits in between acting as the interface layer. In general, the ADC and the bit-width of the DSP represent significant contributions to the total power of the system. In a transform image sensor, matrix-vector multiplications are performed at the transducer and in the analog domain in order to reduce the ADC and DSP bit-width and computational overhead. The basic operation the computational image sensor is:

$$Y_\sigma = A^T P_\sigma B \tag{49}$$

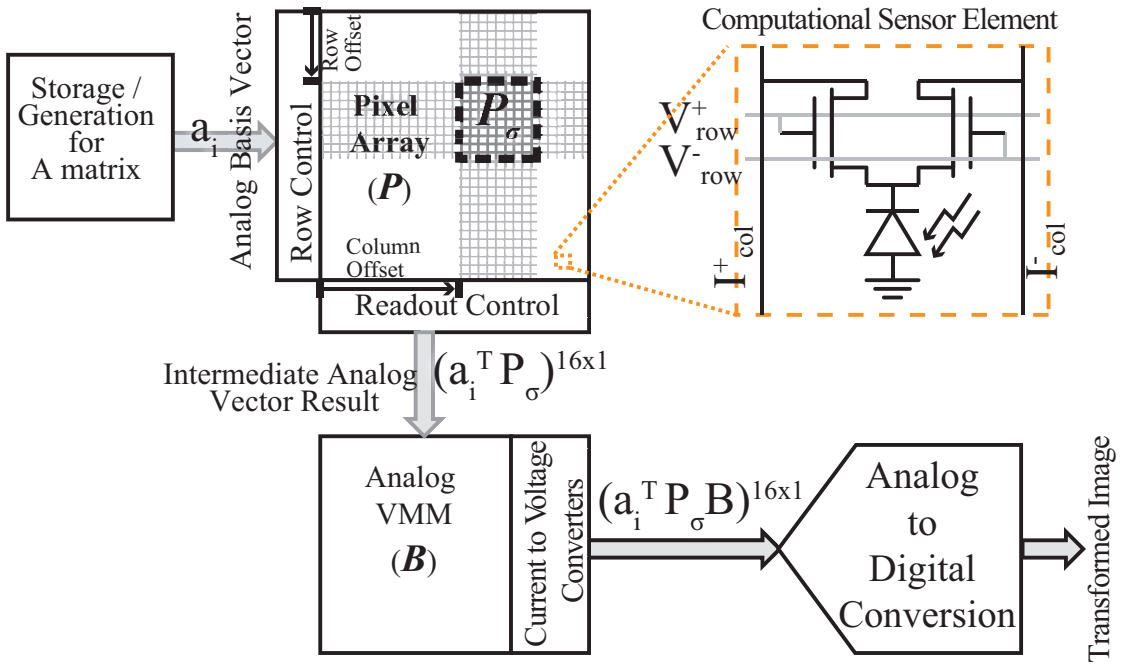


Figure 44. Transform imager system. System diagram of the block transform computational image sensor.

where A and B are transformation matrices, Y is the output, P is the image. The subscript σ denotes the sub-region of the image under transform. A graphical representation of the system is shown in Figure 44.

The first computation is performed at the focal plane, in the pixels, using a computational sensor element shown in Figure 44(b). It uses a differential transistor pair to create a differential current output that is proportional to a multiplication of the amount of light falling on the photo-diode and the differential voltage input. This operation is represented in Figure 45 as the element for the P_σ block.

When the electrical current outputs from a column of pixels are connected, an automatic summation of current occurs. This aggregation results in a weighted summation of the pixels in a column, with the weights being set by the voltages entered into the left of the array. With a given set of voltage inputs from a selected row of A, every column of the computational pixel array computes its weighted summation in parallel. This parallel computation is of key importance, reducing the speed requirements of the individual

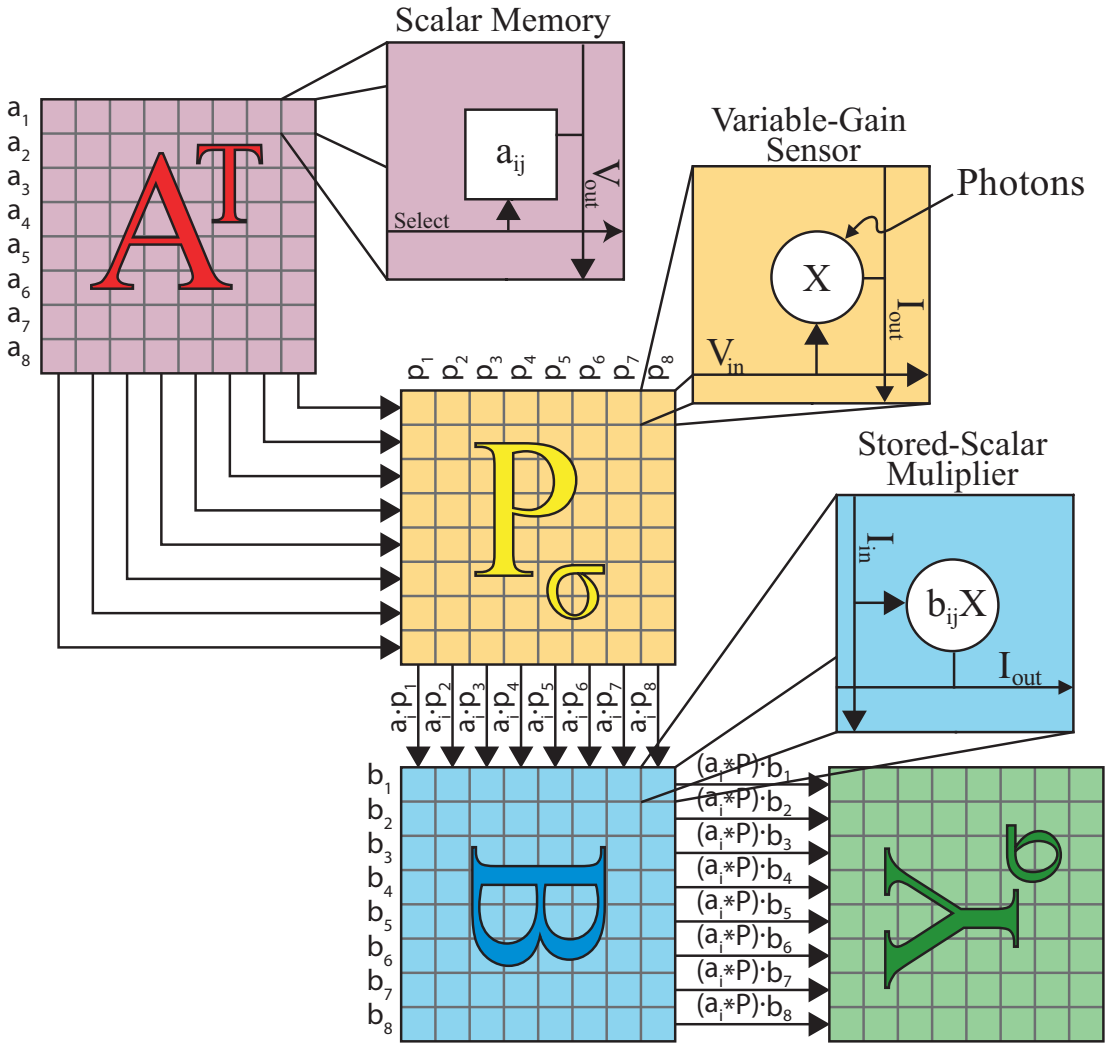


Figure 45. Block matrix computation performed in the analog domain. Illustrated here as an 8×8 block transform, both a computational pixel array and an analog vector-matrix multiplier are used to perform signal projection before data is converted into the digital domain.

computational elements.

The second computation is performed in an analog vector-matrix multiplier (VMM) [30]. This VMM may be designed so that it accepts input from all of the columns of the pixel array, or it can be designed with multiplexing circuitry to only accept a time-multiplexed subset of the columns. This decision sets the support region for the computation. The implementation used for these experiments uses the time-multiplexed column option. The elements of the VMM use analog floating-gate transistors to perform multiplication in the analog domain. Each element takes the input from its column and multiplies it by a unique, reprogrammable coefficient. The result is an electrical current that is contributed to a shared row output. Using the same automatic current summation as the P matrix, a parallel set of weighted summations occur, resulting in the second matrix operation.

6.1.1 Computational Pixel Array

Figure 44 shows a schematic of a single pixel. Each pixel is a photo-sensor and a differential transistor pair, providing both a sensing capability and a multiplication. The output of each pixel is a differential current and it represents a multiplication of the light intensity falling on the photo-sensor by a weighting value represented by a voltage input.

Pixels along a given row of the image plane share a single differential voltage input, which sets the multiplication factor for the row. Pixels along a column share an output line, utilizing KCL to perform current summation. Within each tile is a switch which selectively allows the pixels in the tile to output to the column. When deselected, the pixel currents are switched off of the column's output line to a separate fixed potential.

6.1.2 Random Access Analog Memory

A compact analog memory structure was used to implement a storage for the A matrix, Figure 46. It uses analog floating gates to store the coefficients of the transform matrix, which means that no digital memory or DACs are required to feed the analog weighting

coefficients to the computational pixel array. The use of several DACs along with digital memory would be costly in size and power. Building the memory storage element into the voltage generation structure avoids unnecessary signal handling and conversion, saving size and power.

The basic structure of the analog memory is an amplifier connected as a follower, Fig 46(a). However, one of the differential pair transistors has been replaced with a reprogrammable bank of selectable analog floating-gate pFETs (fg-pFET), Fig 46(b). Each fg-pFET shares the same input V_{bias}, but is programmed to a particular voltage offset which sets the desired output voltage. The programming procedure inherently avoids issues of voltage offsets due to mismatches in the transistors and in the op amp itself by directly monitoring the output in the programming cycle. [24] discusses the use of fg-pFETs, which act much like pFETS that have a programmable threshold voltage offset. Generating 16 differential outputs requires 32 amplifier structures. The storage of a 16×16 differential values requires a total of 32 rows × 16 columns of floating gates. Stacking the amplifiers atop each other creates a 2-D array of floating-gates in a convenient structure for parallel addressing and fits well into floating-gate array programming schemes.

6.1.3 Current Based Vector Matrix Multiplication Design

The back end circuitry of the imager was designed to handle the large line capacitances and high dynamic range of the pixel array's output. Figure 47 shows logarithmic transimpedance amplifiers on the left, which sense and logarithmically convert the pixel current to a voltage. The transfer characteristic is given by

$$\Delta V_{out} = \frac{1}{1/U_t + 1/AV_A} ln\left(\frac{I_{in}}{I_p'}\right) \approx U_t ln\left(\frac{I_{in}}{I_p'}\right), \tag{50}$$

where U_t is the thermal voltage, A is the open-loop gain of the amplifier, V_A is the Early voltage of the feedback transistor, and $I_p^{'}$ is a scaling current. The log is made possible by the subthreshold exponential voltage to current relationship of the feedback MOSFET, much like a BJT or diode implementation [39]. The internal amplifiers, with labeled gain

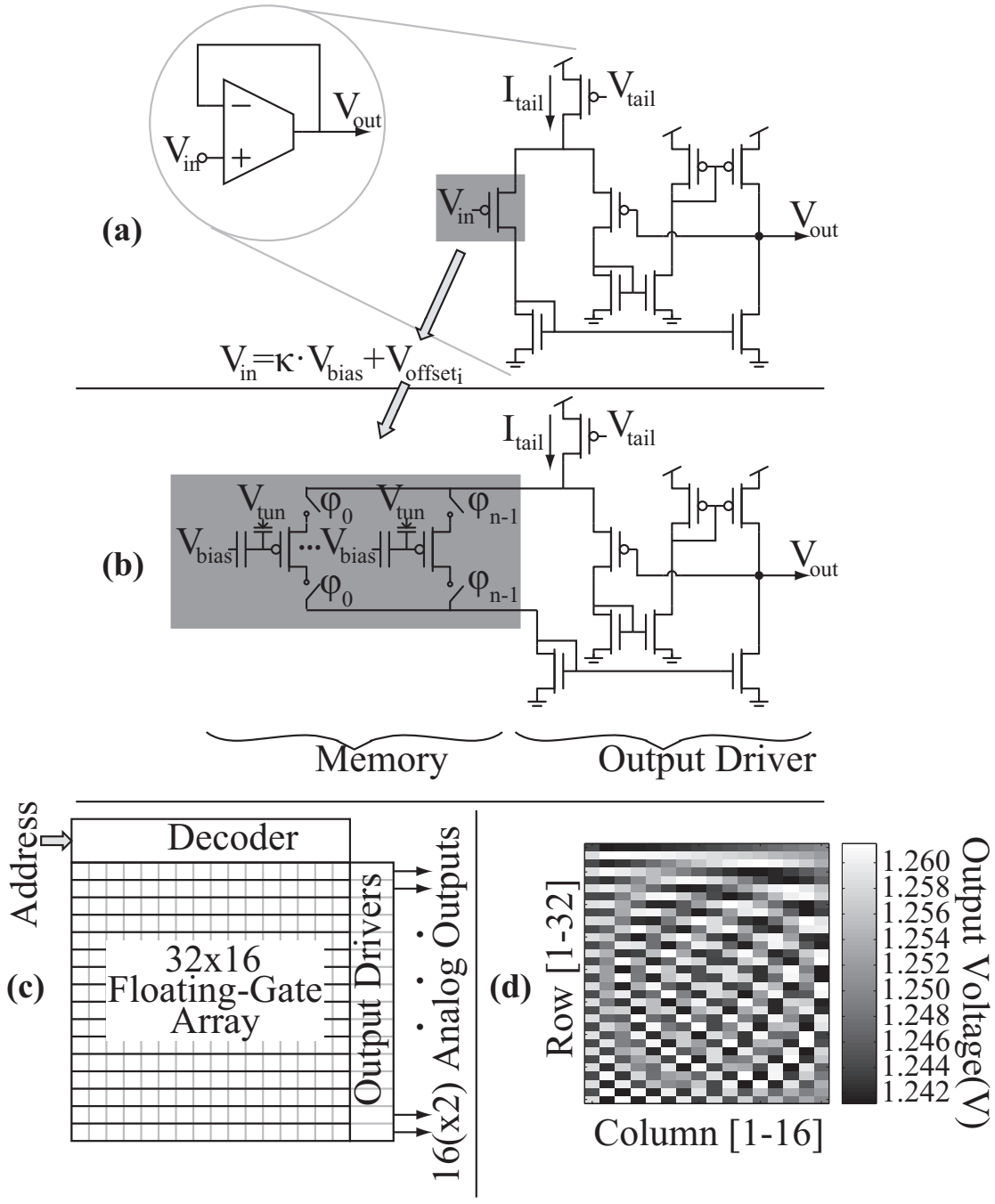


Figure 46. Front-end analog memory for the imager. (a) Basic voltage buffer. (b) Input transistor replaced by selectable analog float-gate transistors. (c) Full analog memory bank. (d)DCT programmed as differential pairs. The differential errors were within 400 uV, approximately our measurement precision.

A, serve a dual purpose: they buffer the outputs of the converter, providing the current for the load transistors, and they create a large loop gain, fixing the input voltage. In addition, they lower the effective input impedance seen at the drain of the feedback transistor from $1/g_s$, where g_s is the subthreshold source conductance of the fg-pFET, to $1/Ag_s$. This low impedance generation is critical to sensing low currents in the presence of large capacitance. The amplifiers can even be matched by programming the fg-pFET.

Unfortunately, the power consumption of this topology is roughly proportional to the dynamic range the circuit is designed to support. This stems from the need to maintain stability in the feedback loop [40]. Since the dynamic range is several orders of magnitude, significant costs are incurred in order to support the full range. To alleviate this, an automatic gain control (AGC) amplifier was integrated into the feedback loop, reducing power consumption dependence on dynamic range support. This is also discussed in [40]. Since subthreshold source conductance, I/U_t, scales with input current, the gain A can be allowed to drop with higher input currents while still maintaining the low input impedance and stability. The AGC amplifier lowers its gain at higher output voltages, which correspond to larger input currents.

The log amp plays an integral role in the analog vector matrix multiplier (VMM), which performs the B matrix multiplication. As shown in Figure 47, every fg-pFET in the array, coupled with the respective row's log amp, forms a wide range, programmable gain current mirror. The current mirror utilizes the sources of the transistors for signal propagation instead of the gates, as in [30], minimizing power law errors due to mismatches in gate-to-surface coupling. Each quadruplet of VMM fg-pFETs corresponds to one coefficient in B. For a fully differential multiplication, w, the programmed gains for a quadruplet are set to $\begin{bmatrix} 1 + w/2 & 1 - w/2 \\ 1 - w/2 & 1 + w/2 \end{bmatrix}$. All VMM transistors along a row share the same input signal and perform their respective multiplications. The output currents are summed along the columns. The resulting differential current output vector is a vector-matrix multiplication

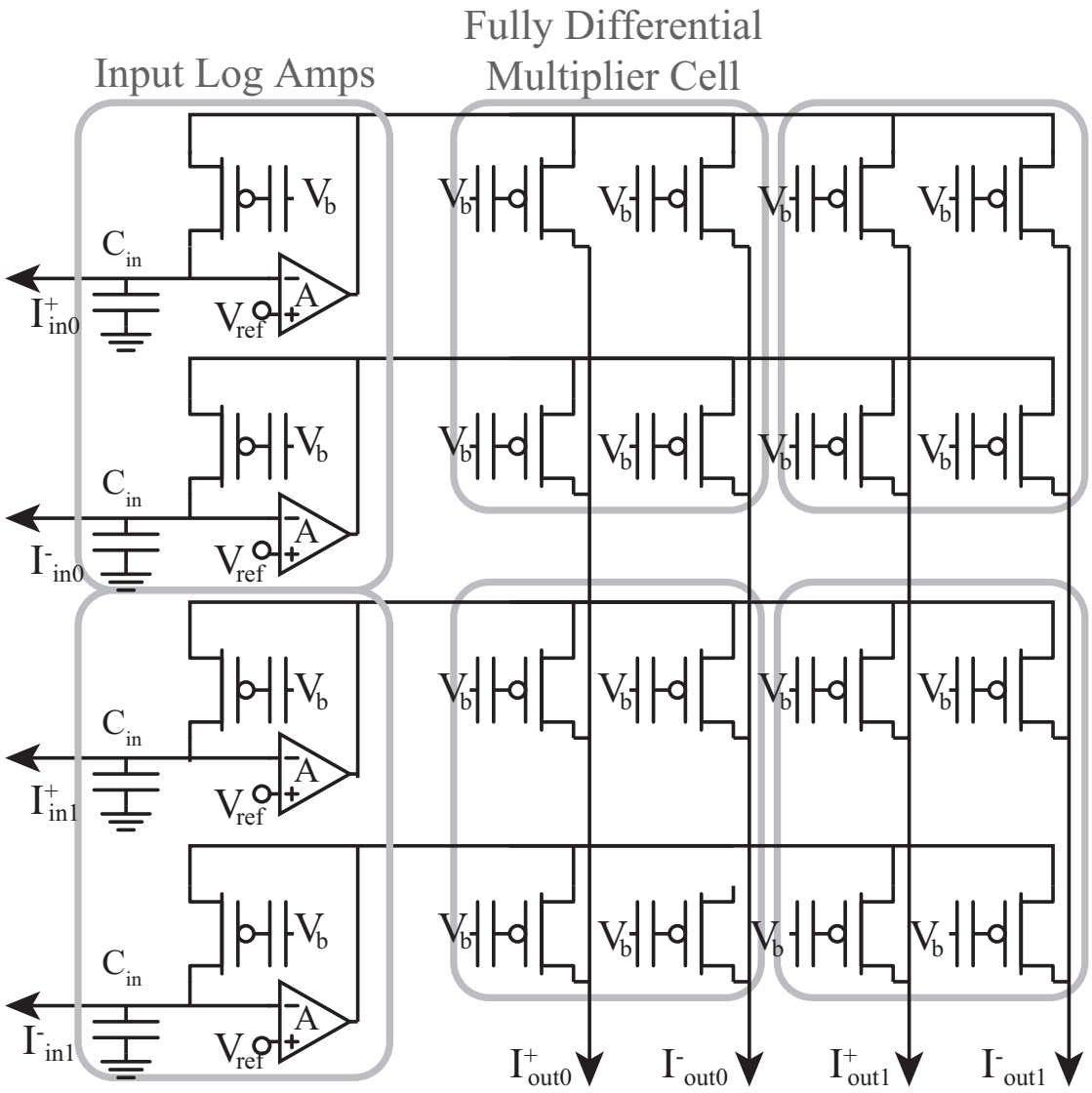


Figure 47. Vector-matrix multiplier schematic. Every fg-pFET in the array, coupled with the respective row's log amp, forms a wide range, programmable gain current mirror. The current mirror utilizes the sources of the transistors for signal propagation in order to minimize power law errors due to mismatches in gate-to-surface coupling. Each quadruplet of VMM fg-pFETs corresponds to one coefficient in B. For a fully differential multiplication, w, the programmed gains for a quadruplet are set to $\begin{bmatrix} 1+w/2 & 1-w/2 \\ 1-w/2 & 1+w/2 \end{bmatrix}$.

vB. The work in [17] implements a similar VMM structure without the automatically-set, high gain necessary for reducing power consumption and supporting wide dynamic range.

6.1.4 Log Bidirectional Current to Voltage Conversion

Since the output of the VMM is a differential current, a differential to single-ended conversion was required. With the desire to maintain the ability to process wide dynamic range signals, a logarithmic conversion was sought. Because the resolution of a logarithmic signal representation is proportional to the signal, it is desirable to remove the common-mode component of the signal before the conversion. This can be achieved by taking the difference of the signals in differential pair. The problem with this approach is that the resulting current can be a very small, even zero. This presents a difficult scenario for a logarithmic amplifier, whose speed is proportional to the input current. Furthermore, a bidirectional logarithmic converter is required. Solving these issues entailed creating a new topology for a bidirectional converter. The design is derived from the mentioned designs of the unidirectional logarithmic converters. Adding bidirectional capability to logarithmic converters typically entails two feedback paths, one for positive current and one for negative. The problem is a dead zone created at near zero current, where neither feedback path is effectively working. By looking at a simple two-transistor I-V converter that incorporated a bias current that had a very useful DC response, a new topology was created. The circuit in Figure 48(b) has nearly the same I-V conversion characteristic as that in Figure 48(a), though the sign is negated and an asymmetry between positive and negative currents is introduced form the followers' feedback: κ_{pf} vs. κ_{nf}. The complete transfer characteristic is

$$I_{in} = I_b \left(e^{\frac{\kappa_{pf} \Delta V_{out}}{U_t}} - e^{\frac{-\kappa_{nf} \Delta V_{out}}{U_t}} \right) \tag{51}$$

It can be approximated by a two-part piecewise function when enough current is being converted:

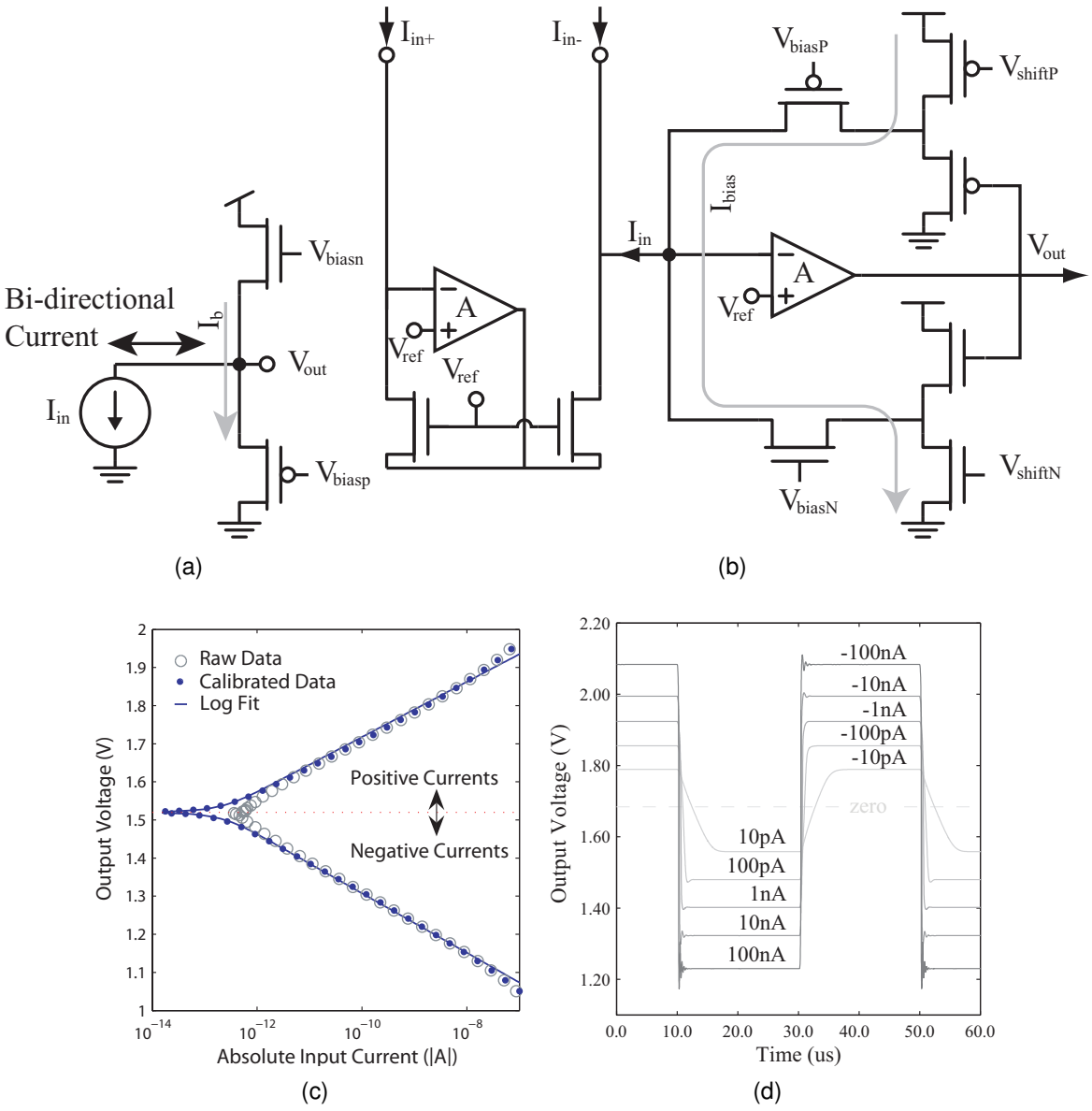


Figure 48. Bidirectional I-V concept and implementation. (a) Simple Compressive I-V (b) High-speed, low-current differential-to-single-ended I-V converter (c) data showing DC characteristic of bidirectional I-V (d) Simulated Bidirectional I-V Converter Step response

$$I_{in} = \begin{cases} I_b \left(e^{\frac{\kappa_{pf} \Delta V_{out}}{U_t}} \right) & \Delta V_{out} \gg 0 \\ I_b \left(-e^{\frac{-\kappa_{nf} \Delta V_{out}}{U_t}} \right) & \Delta V_{out} \ll 0 \end{cases} \tag{52}$$

The new topology utilizes voltage offsets, in this case two-transistor followers, to establish source voltages on the feedback transistors that move in response to the main amplifier's output and maintain a bias current through the transistors, guaranteeing a minimum input conductance and therefore speed even when the input current is zero. The DC response, logarithmic in nature, supports a wide range of signals while maintaining good resolution throughout the range, Figure 48(c). This structure also utilizes a AGC amplifier, which in this case loses gain as the output deviates from the zero-current output voltage.

To perform the precursory current subtraction that converts the differential signal to a single-ended signal, a current mirror, utilizing the source node for signal propagation as in the VMM, is used. Though a gain error may occur due to threshold voltage mismatch in the current mirrors, this is accounted for when programming the corresponding column of the VMM.

Thus, as the input current deviates from zero, the converter approximates a logarithmic compression. This bi-directional converter is very useful in applications where support for large dynamic range is essential and small currents must be sensed at bandwidths well beyond g_m/C. As in the unidirectional structure, the internal amplifiers boost the bandwidth in proportion to their gain.

6.1.5 Test Setup

In order to test our computational image sensor, we constructed the instrumentation platform pictured in Figure 49a. On the computer we used Matlab as a high-level interface, which connected to the FPGA board through a full-speed USB-parallel converter from FTDI. In order to achieve a maximum bandwidth of 6 $Mbps$, we constructed a Matlab executable which interfaced to the custom FTDI D2XX driver in C. We selected an FPGA board from Microtronix which contained an Altera Stratix chip. The core processing work

on the FPGA was controlled by Altera's Nios II 32-bit RISC soft processor. One of the key features of the soft proessor was an extensible 32-bit bus called Avalon. We implemented custom Avalon slave interfaces for all of the hardware on the FPGA in order to have a direct memory-mapped interface available in the software environment on the NIOS processor. By creating a simple messaging protocol on top of the USB communication, the memory-mapped interface was extended to Matlab. By coupling the high-level memory-mapped interface with the queueing capability of the D2XX driver, large control sequences for the hardware could be predetermined and dispatched to the CPU at a rate communsurate with implementing the sequences directy in C on the soft processor–this allowed for algorithmic prototyping with all of the advantages of a powerful, interpreted language (Matlab), with less instrumentation overhead. The CPU did not run at a high enough speed to dispatch digital control sequences to the imager IC, so a light-weight multi-cycle support processor was constructed to faciliate digital control. The external RAM was used to accumulate data when the bandwidth to the PC was not sufficient, e.g., high framerate video capture.

The illustration in Figure 49b represents the method of automating the optical scene presentation to the imager. A programmable light source was used to illuminate a scene on an LCD connected to a video output from a computer. The setup from Figure 49a was mounted on the XYZ translator, and the scene was focused onto the imager.

6.1.6 Results

Technically our imager does not output images, but the inner-products of the projections of the image onto basis sets programmed into the transformation matrices A and B. The measured data from the chip is a hyperbolic sine-like transformation of the actual projection. Each point in the output matrix is a log-compressed voltage representation of the current-mode transform coefficient. The data is related back to the current through the pre-characterized I-V transfer curve, Figure 48c.

In order to measure the image incident with our pixel plane, we employed identities as our transformation matrices. In this way the output of the imager is a representative

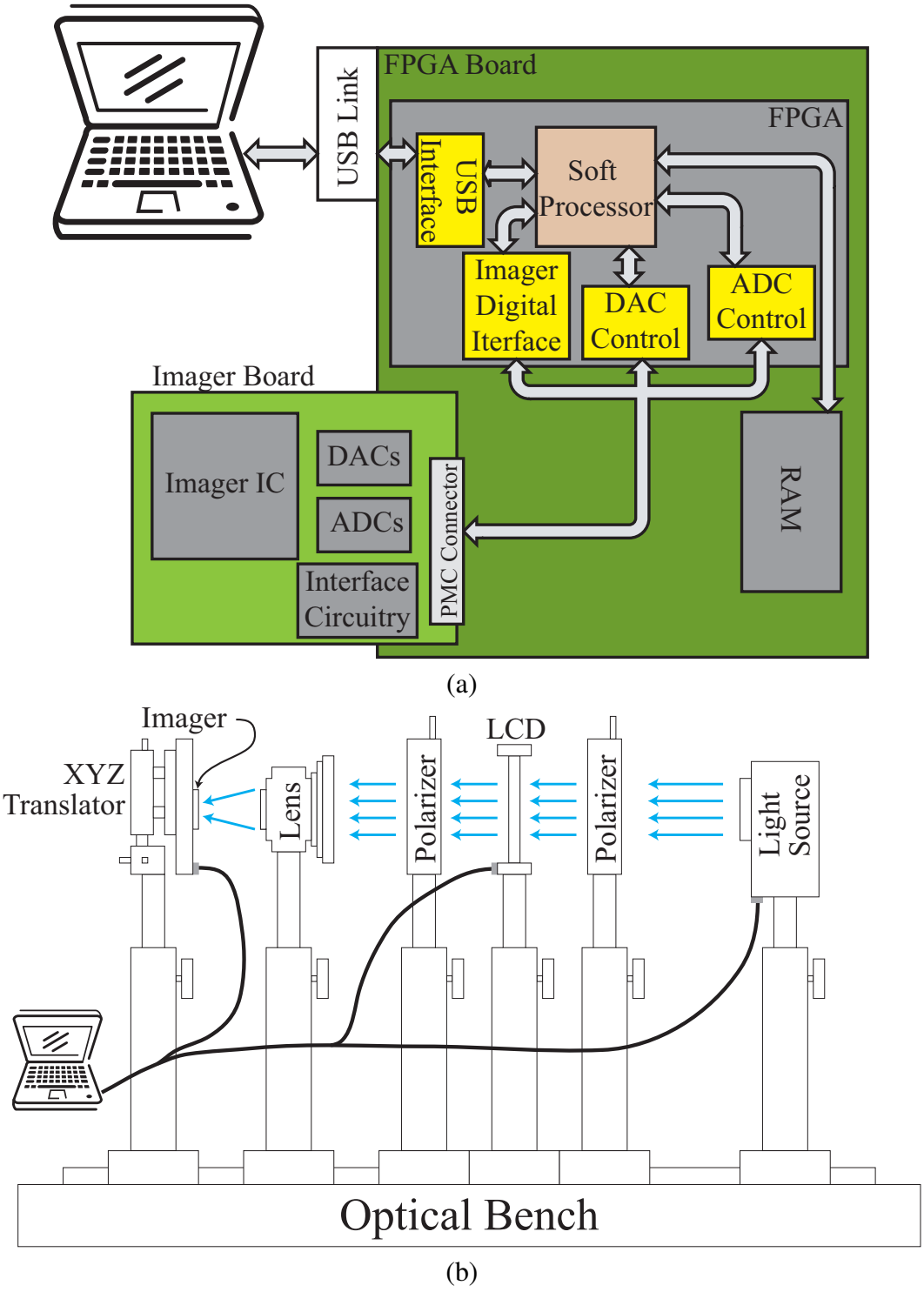


Figure 49. Imager test setup. (a) Instrumentation platform for the imager IC. (b) Optical setup for imager with computer control.

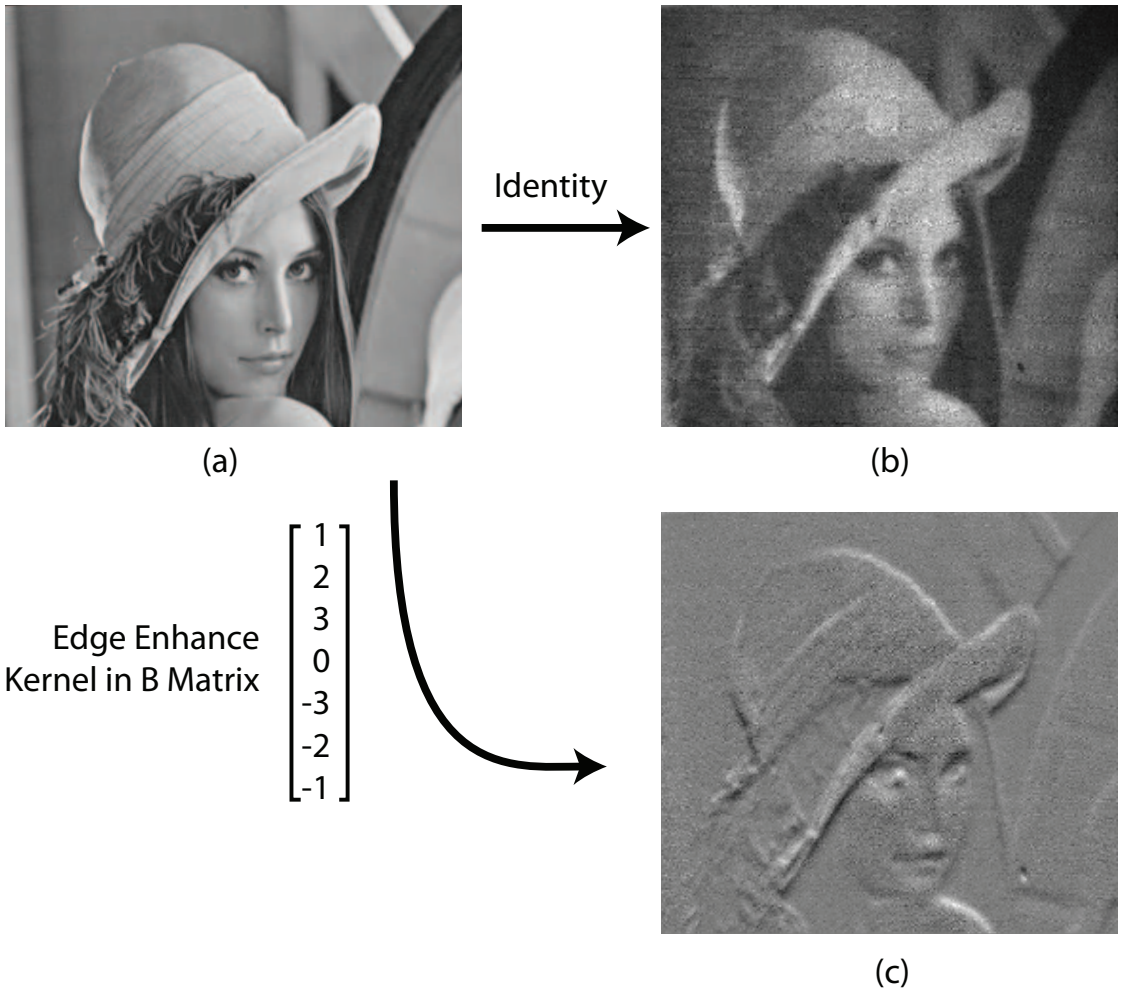


Figure 50. Image reading with identity and high-pass convolution programmed into the *B* matrix. (a) Test image, directed at the image sensor by projection through an LCD. The contrast of the output was limited by the contrast of the LCD. (b) Result of projected image transformed with an identity in both dimensions, and then transformed from voltage to current using Figure 48c. (c) An edge enhancement was performed by programming a high-pass kernel into the *B* matrix.

measurement of the raw image. The test image in Figure 50a was displayed on an LCD and directed at the image sensor, Figure 49. The resulting data that was read out is shown in Figure 50b, post voltage-to-current conversion. We then show an implementation of edge enhancement on the imager by using a 1-D truncated pyramid mask in the *B* matrix, Figure 50c.

By implementing separable transforms that result in sparse representations of the input

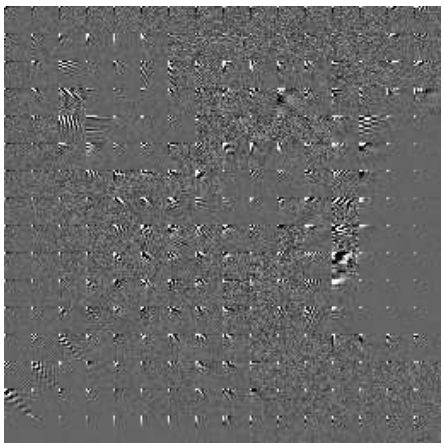
image and combining it with a thresholding operation, analog compression can be implemented. A scene of passive electrical components was first imaged using identities, Figure 51a. Next, we used the discrete cosine transform (DCT) and Haar wavelet transform to process the scene. The DCT is ideal for implementing analog front-end JPEG compression, while the Haar transform was chosen to show the flexibility of our computational image sensor. As seen in Figure 51b and 51d, the data from the IC image is a transformed representation of the image in 51a. Figure 51b shows a DCT transformation result and Figure 51d shows a Haar transformation result. These data sets were processed with ideal inverse transforms to produce Figure 51c and 51e. The focus of the experiment was to create an analog output with a sparse representation, the actual thresholding circuitry was not included on this chip.

One of the particularly desirable aspects of the DCT is that the lower frequency coefficients, where most of the signal power generally lies, are clustered into a local region. That, combined with the separable nature of the transform, significantly reduces the effort of choosing which coefficients to propagate to the next stage of processing. The fidelity cost of DCT compression is in its representation of edges. Coefficients that relate to higher frequency are the ones pruned as they tend to have less signal power, but edge data is lost since edges span low and high frequencies.

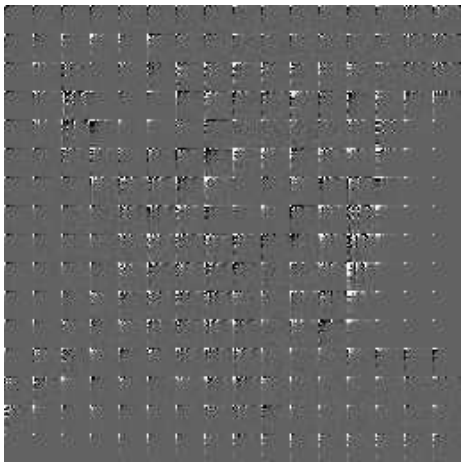
Under most circumstances, wavelets do a better job than the DCT for image compression, since the basis functions implement a high-pass and low-pass at many different scales, but the projection results in scene-specific clustering of high signal-power coefficients. As a result, it is not clear which coefficients are worth processing. In order to get around this limitation, we can use a compressive sensing approach, where a decorrelated basis set is used to capture a seemingly random combination of the pixel data, and convex optimization is used on the resultant data set to reconstruct the image–compression is just a matter of capturing an arbitrary subset of the total coefficient data.

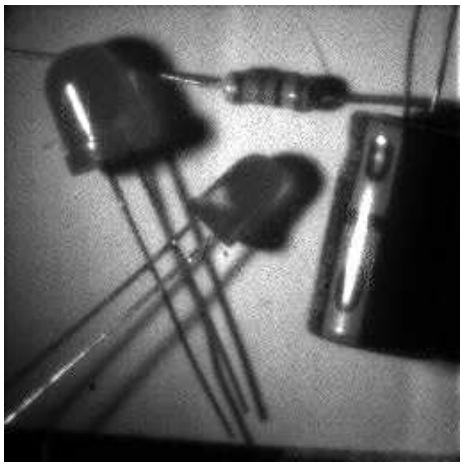
In the experiments, a complete set of transform coefficients were collected, and the

(a)



(b) (c)





(d) (e)

Figure 51. Imaging with transforms that yield sparse representations, DCT and Haar. Both transforms were performed completely in analog on the transform imager. The data was read out in voltage, then converted to current using the characteristic of the readout circuit. (a) Scene imaged with identities for the A and B matrix. (b) DCT transform of the scene. (c) DCT Reconstruction using ideal inverse DCT. (d) Haar transform of the scene. (e) Haar reconstruction using ideal inverse of the Haar transform.

reduced collection was simulated by discarding measured values. The nonlinear recovery algorithm discussed was used to reconstruct the images. Since the exact original image is not available, reconstructed images corresponding to incomplete collection were compared against denoised versions of images created from complete coefficient collection.

At high levels of compression, retaining few transform coefficients, the DCT representation lead to better peak signal-to-noise ratio (PSNR), Figure52a and 52b. This is possible because the predefined DCT coefficient removal process exploits the knowledge of where energy compaction occurs in the DCT domain. In the case of the noiselets, higher transform coefficient retention lead to better performance, surpassing the DCT results in quality. It is expected that every transform coefficient in the noiselet domain statistically contributes the same signal and noise power to the resulting image as any other coefficient. In the case of DCT transform coefficients, the coefficients representing high spatial frequencies contribute the same noise as the coefficients representing low frequencies, but they contribute less signal power. This is believed to be a contributor to the PSNR actually falling in the reconstructed image as more coefficients are added. Additionally, it is believed that the noise in the DCT images are overall higher because the DCT bases are smaller in magnitude than those of the noiselets when implemented in the analog system. The basis functions are constrained to a linear input range of the analog computational elements. Since the noiselet functions consist of only 1's and -1's, they use the complete signal range of the system, resulting in better signal to noise ratio.

The computational image sensor was re-fabricated using the VMM design techniques discussed in Chapter 5. Figure 53 shows some preliminary images taken from that imager. A full frame capture takes $1/3$ of a second, with a single $16x16$ block capture occuring at 50 frames per second. The bottleneck for reading images is in the instrumentation platform.

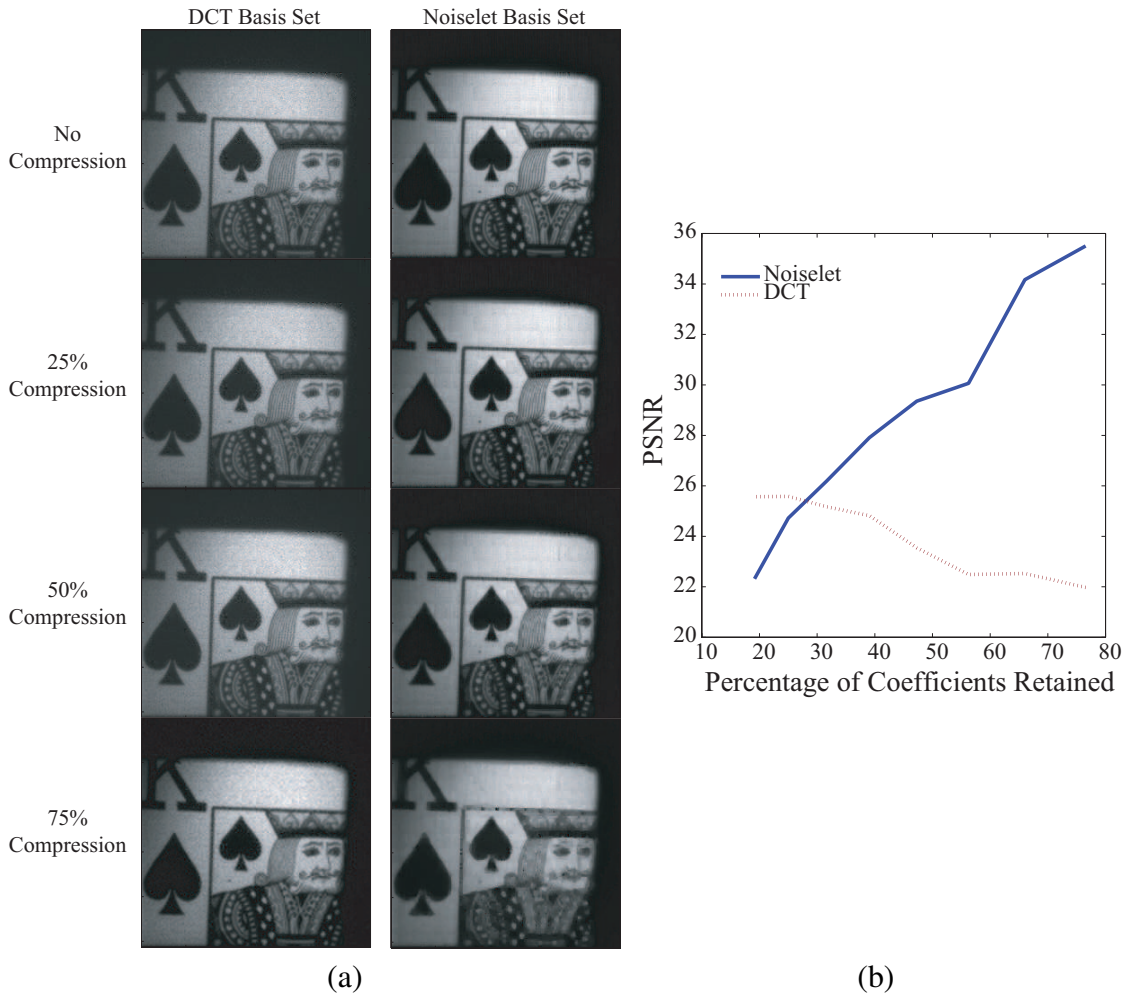


Figure 52. Compressive Sensing using the transform imager. (a) Reconstruction results using DCT and Noiselet basis sets with various compression levels. The image sensor measured 16×16 blocks of the image projected onto DCT and noiselet basis functions. Subsets of the data were taken and used to reconstruct the shown images using a nonlinear total variance minimization algorithm for the noiselets and a pseudo-inverse for the DCT. (b) PSNR of reconstruction vs. percentage of used transform coefficients.

6.2 Adaptive Filter

This work demonstrates a fully integrated, compact, adaptive filter layer based upon a continuously adapting node. These nodes adapt through the Least-Mean-Square (LMS) learning algorithm based upon continuously adapting floating-gate circuits. Figure 54a shows the block diagram of an adaptive linear combiner; the feedforward computation is

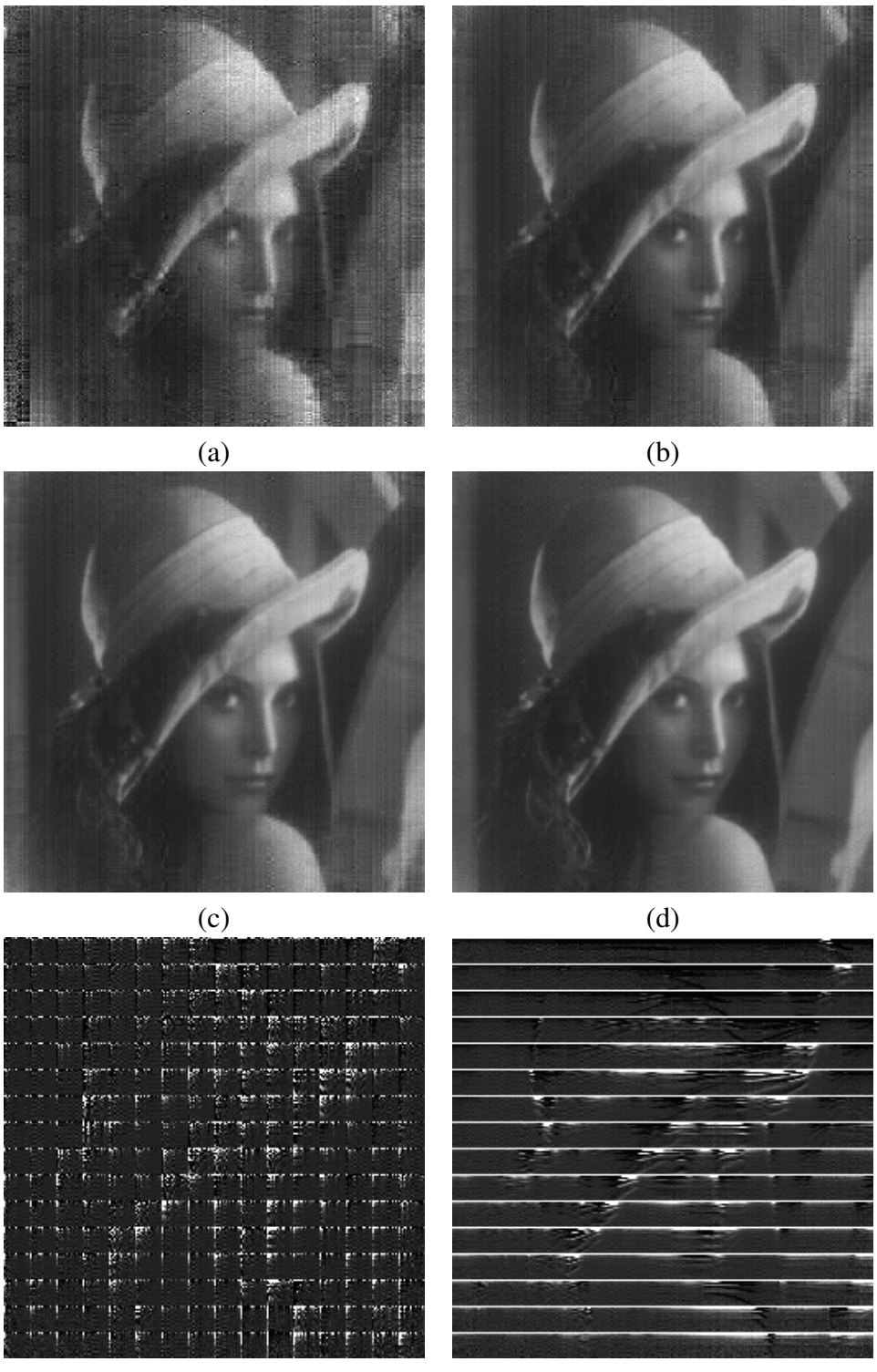


Figure 53. Different levels of averaging on the imager for a full frame, 256x256. The framerate is limited by the instrumentation. The first image is a HAAR reconsctruction with no averaging, while the subsequent images were averaged 10, 25, and 100 times and then reconstructed using an ideal inverse HAAR, Figures 53b–53d. The framerates were 3.3, 1.8, 1, and .33 frames per second. Figures 53e and 53f are the raw and partial HAAR reconstruction of 53d.

described as

$$y = \sum_{j=0}^{n-1} w_j x_j \tag{53}$$

where, x_j is the j^{th} input and w_j is the stored weight at position j. Applications for adaptive filters include adaptive equalization, prediction, system identification and noise cancellation.

Figure 54a shows the implemented system architecture. Exploiting the non-linearities inherent in hot-electron injection and Fowler-Nordheim tunneling, floating-gate transistors adapt the weight level along an LMS rule [41]. Weight adaptation is obtained by comparing the sum of the outputs of multiple synapses to a desired target and changing the weights of each synapse such that the error between the target and the system output is minimized. Using floating-gate transistors for both weight adaptation and weight storage results in the synapse circuits being compact and low-power [41, 42]. The use of floating-gate transistors provides a non-volatile storage capability for the weights. The proposed analog architecture has been fabricated in a $0.35\mu m$ CMOS process (die photo in Figure 54c).

Adaptive filter design in the analog domain is motivated by the benefits of a low-power implementation. Digital multiplication, addition, and integration are both power and area intensive while an analog approach can be both compact and power efficient [43]. Although analog adaptive filters have been implemented previously ([44, 45, 46, 47, 48]), the floating-gate approach combines weight storage, feedforward multiplication, and weight adaptation in a compact and efficient manner.

6.2.1 Adaptive Filter Architecture

The block diagram of the analog adaptive filter system is shown in Figure 55a. The system implements the adaptive linear combiner as described earlier. Each adaptive node consists of 16 synapse elements and the chip contains 4 nodes. A total of 16 analog input signals can be provided with the signals being shared between all four adaptive nodes while the target signal is unique for each node. The output of each node is compared with the target

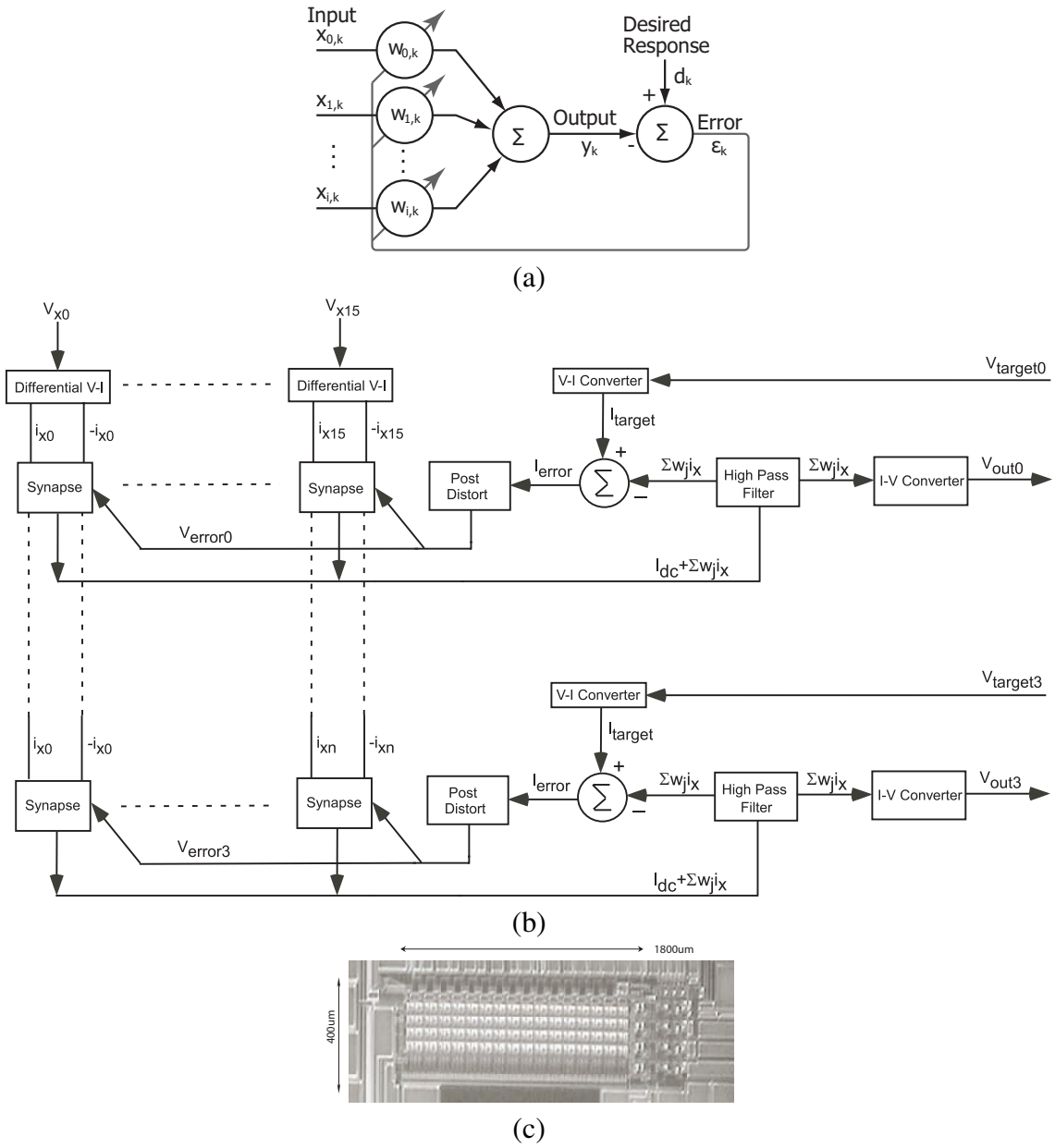


Figure 54. Our adaptive filter IC. (a) Block diagram representation of an adaptive filter / linear combiner that adapts its weights such that the error between its output and the target signal is minimized. (b) Block level representation of an analog adaptive filter composed of multiple adaptive nodes. (c) The die photograph of the fabricated adaptive filter system in a $0.35\mu m$ CMOS process.

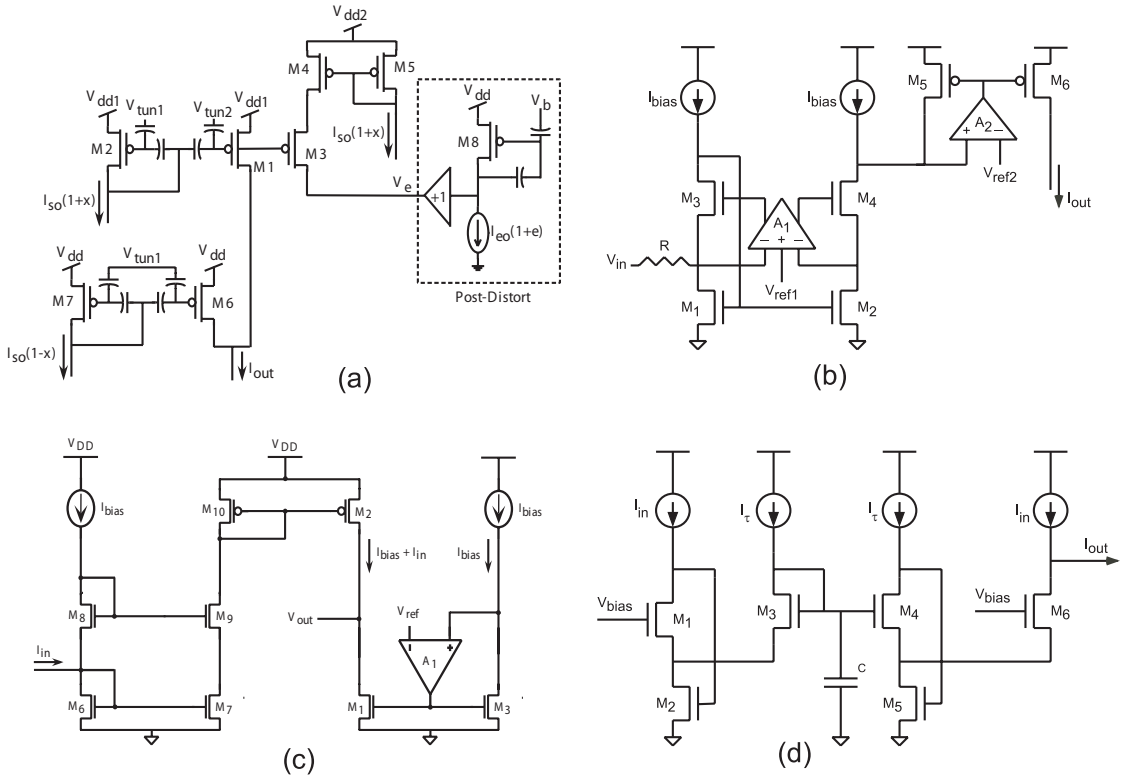


Figure 55. Adaptive filter circuit schematics: (a) Floating-gate based synapse circuit along with the post-distort circuitry that is common to all the synapses in a particular row. (b) Circuit schematic of the voltage-current converter. (c) Circuit schematic of the current-voltage converter. (d) Circuit schematic of the log-domain current-mode high-pass filter.

signal and the error signal is fed back to each of the synapses in a given node. The weights of the synapse adapt such that the error signal is minimized and the output of each node tracks the target signal.

The system employs current-mode signaling with current-voltage (I-V) and voltage-current (V-I) converters forming the interface. Figure 55b shows the circuit schematic of the V-I converter [49]. The use of a regulated cascode current mirror ($M1 - M4$) ensures that the drain of $M1$ is set to a well defined value of V_{ref}, independent of the current flowing through $M1$. The use of amplifier $A2$ helps fix the output node at a fixed voltage thereby nullifying the effect of the output capacitance leading to a high bandwidth. Figure 55c shows the circuit schematic of the I-V converter [49] where transistor $M2$ is a common source amplifier with $M1$ being the active load to perform the current conversion. The issue of stabilizing the high gain output node is addressed using the replica transistor $M3$, identical current source I_{bias} and the amplifier $A1$. Figure 55d shows the log-domain current-mode high-pass filter schematic. Transistors $M1 - M6$ low-pass filter the input signal. The input signal is encoded at the source of $M1$ and is low-pass filtered at the gate of $M3$. Transistors $M4$, $M5$ and $M6$ take the low-passed gate voltage of $M3$ and transform it into a low-passed version of the input signal. The low-passed signal is subtracted from a copy of the input signal at the drain of $M6$ resulting in I_{out}, a high-pass filtered version of the input.

Characterization results for the circuit blocks are shown in Figure 56. Figure 56a shows the DC transfer characteristic of the V-I converter that displays a linear range of about $3.5\mu A$ at a bias current on $5\mu A$. Figure 56b shows the DC transfer function for the I-V converter. The slope is negative because of the sinking nature of the input. The measured transimpedance gain of the I-V converter is about $1.6M\Omega$. Figure 56c shows the effective bandwidth of the system to be $\approx 110KHz$ by driving the I-V converter using the V-I converter. Figure 56d shows the step response of the high-pass filter, where the input to the

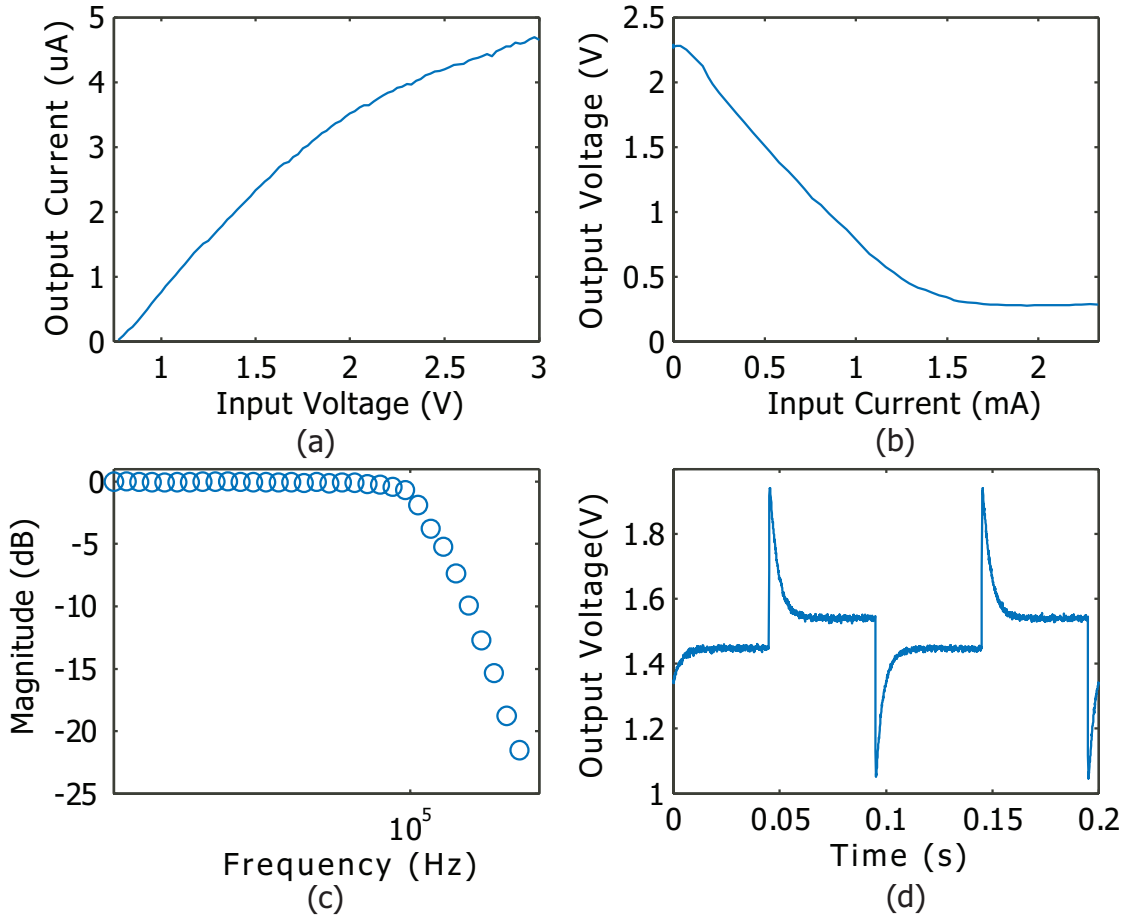


Figure 56. Characterization of Functional Blocks Required to build the adaptive nodes. (a) DC transfer function of the *V*-*I* converter. (b) Transfer characteristics of the *I*-*V* converter. (c) Frequency response of the combination of *V*-*I* and *I*-*V* converters. (d) Response of the high-pass filter to a series of steps applied at its input.

filter is applied using the *V*-*I* converter and the current output is read using the *I*-*V* converter. The filter displays a high-pass behavior with an offset due to the mismatch between transistor pairs $M1/M6$, $M3/M4$ and current mirror mismatches.

6.2.2 Adaptive Synapse Operation

The floating-gate synapse [41] that implements the least-mean-square learning rule is shown in Figure 55a. Transistor pairs $M1/M2$ and $M6/M7$ implement a differential synapse such that both positive and negative weights can be realized. It should be noted that adaptation occurs only at the floating-gate of $M1$ while the floating-gate of $M6$ is programmed to an

equilibrium weight that acts as a reference. The drain currents of $M1$ and $M6$ are summed together and become the synapse output. The drain current of $M1$ is given by,

$$I_1 = I_{so}(1 + w)(1 + x) \tag{54}$$

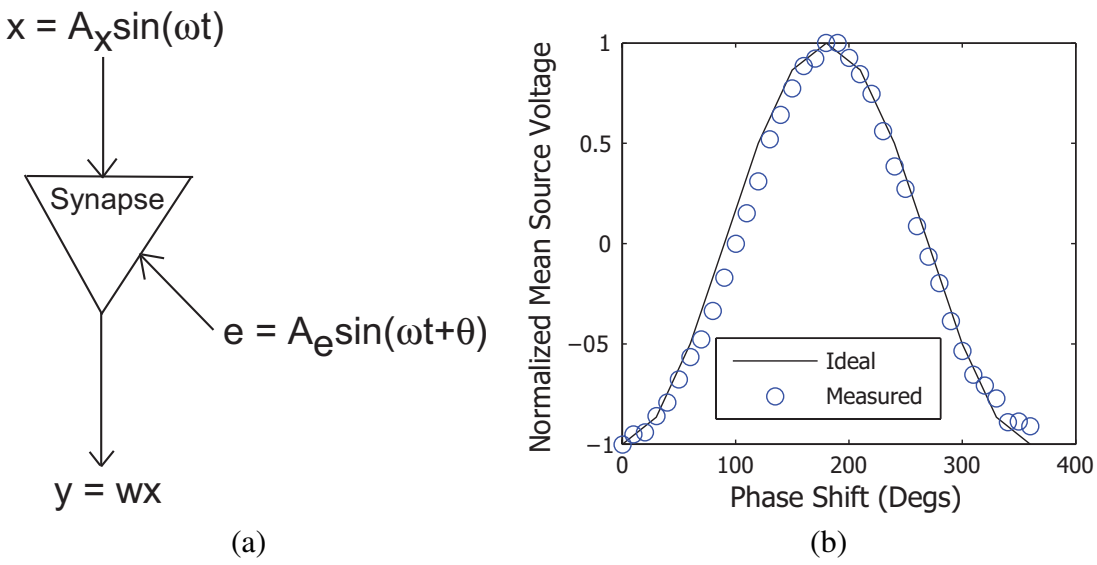
where I_{so} is the equilibrium bias current. The above equation has been derived assuming that transistors $M1$ and $M2$ are identical, their input capacitors are matched. Performing a similar analysis for transistor $M6$ and high-pass filtering the resulting sum of the drain currents of $M1$ and $M6$ gives the required synapse output current, $I_y = wx$, where w is the difference between the weight adapted on $M1$ and the equilibrium weight stored on $M6$.

The weight adaptation in the synapse is achieved using the physical phenomenons of tunneling and hot-electron injection. Transistor $M3$, that forms a source-follower along with transistors $M4 - M5$, aids in the process of adaptation. The error signal for adaptation is fed back via the post-distort circuitry. The post-distort circuitry shown within the dotted lines in Figure 55a performs a non-linear conversion of the error current into a voltage that when applied to the drain of transistor $M1$ results in a linear mapping between the error current and the drain current of $M1$ [41]. Applying KCL at the floating-gate node and modeling the tunneling and injection currents as in [41], the weight update equation is found to be similar to an LMS learning rule and is derived to be [41],

$$\tau \frac{dw}{dt} \approx -(\epsilon - \alpha E[xe])w + E[xe] \tag{55}$$

where τ is inversely proportional to the tunneling bias current, and ϵ is the weight decay parameter set by device parameters (less than 0.1).

Figure 57 shows a characterization of the synapse circuitry. In order to demonstrate the correlation behavior of a single synapse as given in (55), a sinusoidal signal is applied to both the input and the error terminal of the synapse circuit. According to (55), the synapse computes the correlation between the two signals, the result being a change in the DC level of the floating-gate voltage. This change in the floating-gate voltage results in a DC change in the source voltage. For an LMS learning rule, with two sinusoidal signals applied to the



(a) (b)

Figure 57. Results demonstrating correlation behavior by applying a sinusoidal input current and a phase shifted sinusoid at the input of the post-distort circuitry. (a) Phase correlation experiment set-up. (b) Plot of the normalized weight vs. phase shift of the error signal sinusoid.

input and the error voltage, the equilibrium weight is approximately given by,

$$w_{eq} = \approx \frac{A_i A_d}{2} cos\theta \tag{56}$$

where, A_i, A_d are the amplitudes of the applied sinusoidal inputs and θ is the phase difference between the two signals. Measuring the steady-state value of the source voltage for different sinusoidal inputs at the input and the error terminals of the synapse should result in a cosine function. Experimental results are shown in Figure 57b that confirm correlation learning in the synapse.

Figure 58 shows the synapse response with a square wave target and a sinusoid of the same fundamental at the input. The weight peaks when the input and target are maximally positively correlated, 0° phase difference, shown in Figure 58a,b. The scale factor becomes 0 at maximum negative correlation, 180° phase difference, shown in Figure 58c,d. This clearly demonstrates LMS learning in the system.

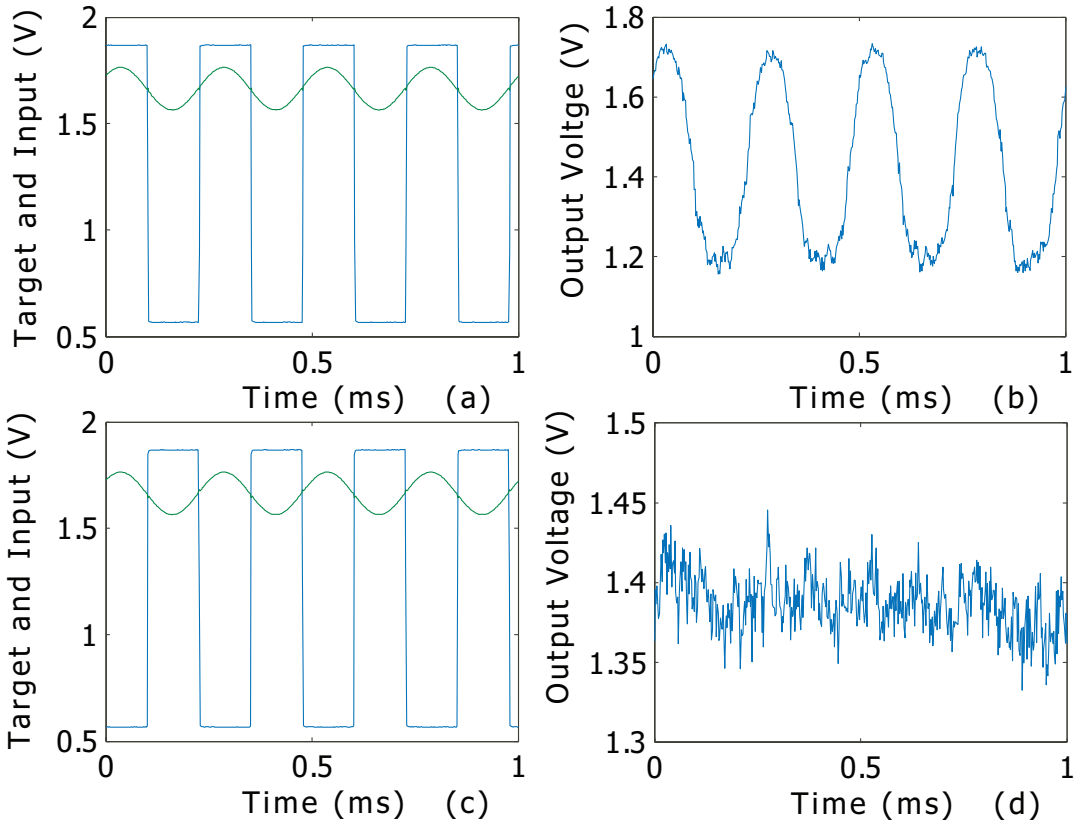


Figure 58. Adaptation of a single synapse connected using an LMS feedback. (a) Target square wave and input sine-wave applied in phase. (b) Output of the synapse adapts to being a sine-wave. (c) Target square wave and input sine-wave applied with 180° phase difference. (d) The equilibrium weight of the system goes to zero therefore the synapse outputs a DC signal instead of a sine-wave.

Table 1. Equilibrium weights for a Fourier Decomposition Experiment

Sine Frequency (KHz)	1	2	3	4	5
Meas. Weight	1	0.0445	0.3142	0.0469	0.1881
Fourier Coefficients	1	0	0.33	0	0.2

6.2.3 Adaptive Filter Measurements from a Network of Nodes

The experimental setup consists of a custom PCB for the chip that contains the hardware necessary for programming floating-gate transistors. The delay lines were implemented off-chip using digital-to-analog converters (DACs) and smoothing filters (low-pass filters) to provide flexibility for testing different applications. The setup is controlled using a FPGA board with a computer in the loop, resulting in a fully automated test fixture. This provides the flexibility of implementing a variety of learning scenarios as arbitrary waveforms that can be generated in software and applied to the chip using DACs. Experimental results that have been measured using the test setup are presented in this section to demonstrate adaptation and learning.

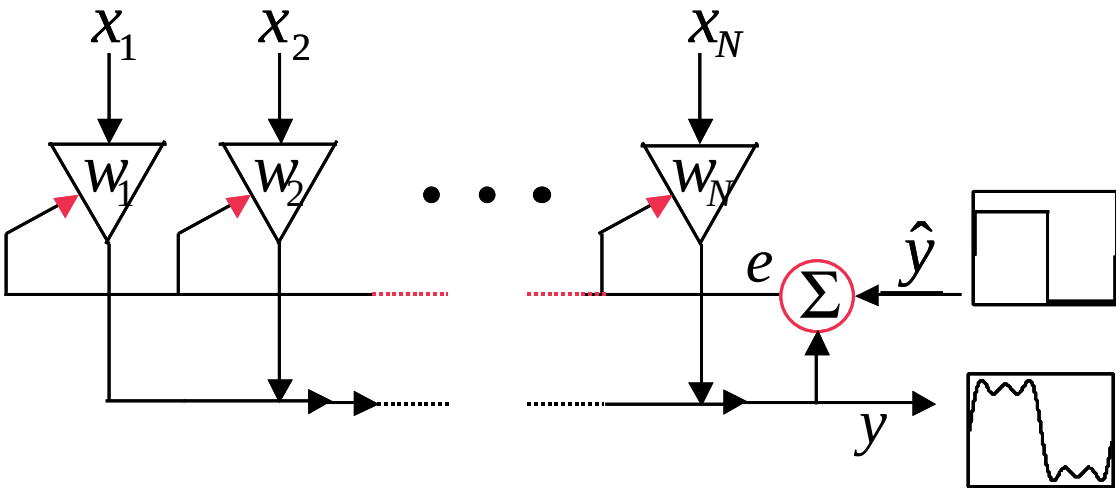
A Fourier decomposition experiment was performed on the synapse; an adaptive linear combiner can learn a square-wave when presented with sinusoids that are at integer multiples of the square-wave frequency. Figure 59a shows this experiment, where the weights adapt to the Fourier co-efficients such that the output resembles a square-wave with the result that the error between the output and the target is minimized. The chip was presented with a $1KHz$ square-wave target and the equilibrium weight was measured by providing the first five harmonics of the target square-wave. The solid line in Figure 59b shows the ideal square-wave that results when the first five harmonics are weighted with the ideal Fourier co-efficient and combined together. The circle data of Figure 59b shows the resulting square-wave using the weights obtained from the chip. As can be observed, a square-wave results when the first five harmonics are combined using the measured weights, thereby demonstrating learning in the chip. Table 1 presents the weights obtained experimentally by conducting the above experiment and compares them with the

Table 2. Summary of Performance

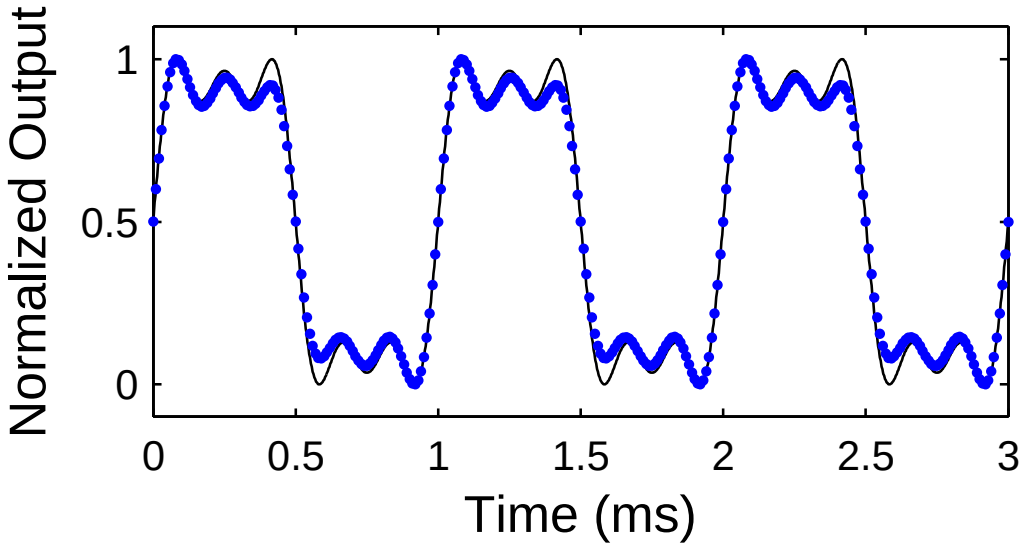
Parameter	Value
Power Supply	$3.3V$
Process	$0.35\mu m$ CMOS
Area	$1800\mu m \times 400\mu m$
Power Dissipation	$13.2mW$
Adaptation Mechanism	Hot-electron injection and Tunneling
Adaptation Time	$1ms - 10s$
Input Signal Bandwidth	$100KHz$

ideal expected value. As can be observed, the weights converge to the ideal values, clearly demonstrating adaptation in the analog implementation.

Table 2 summarizes the key performance parameters of the fabricated adaptive filter chip. The chip occupies an area of $1800\mu m \times 400\mu m$. The chip contains 4 adaptive nodes with 16 synapses each for a total of 64 synapses and associated circuitry. The entire chip dissipates a power of $13.2mW$ at an operating supply voltage of $3.3V$. The bulk of the power is dissipated in the amplifiers and buffers used in the interface circuitry to drive signals on and off the chip. The synapse matrix consumes a power of approximately $633\mu W$. The use of tunneling and injection as the mechanisms controlling adaptation enables adaptation time constants in the range of $1ms - 10s$. The wide range of time-constants that are available is a key advantage in the proposed approach. This makes possible a range of learning problems while other techniques require large capacitors with the time constant being ultimately limited by diode leakage currents. The bandwidth of the system is limited primarily by the bandwidth of the I-V converter.



(a)



(b)

Figure 59. Adaptive node Fourier decomposition. (a) Schematic showing learning a square-wave from harmonic sinusoids (b) Square-wave resulting from normalized Fourier co-efficients for the first five harmonics. The solid line is ideal, the circles represent measured equilibrium weights obtained from the analog adaptive filter chip.

CHAPTER 7

RECONFIGURABLE ANALOG SYSTEMS

In building large-scale reconfigurable analog systems, there is a fundamental choice to be made between the level of reconfigurable and the level of area consumption and associated increase in circuit parasitics. Field programmable analog arrays (FPAAs) are designed with a bias toward reconfigurability. Fundamentally, an FPAA is broken down into elements of connection and elements of computation, though the elements can be chosen in such a way that the computational elements implement connectivity. The salient point is that floating-gate transistors go a long way toward enabling large-scale FPAA systems. The non-volatile analog memory element provides a means for implementing both circuit biases and switches in a compact way. In terms of the granularity of the analog components, which are typically grouped together into a computational analog block (CAB), components of coarser granularity lend themselves to less connectivity, resulting in less parasitics and higher performance. Finer granularity in CAB component choice is desirable for constructing an IC that allows for hardware design exploration without the burden of subsequent IC fabrication. The next two sections illustrate two different FPAAs. The Reconfigurable Analog Signal Processor, or RASP, was built with generic analog design in mind. The Reconfigurable Analog Array of MITEs, or RAAM, was constructed with an intent of building a direct link between mathematical equations and circuit implementations.

7.1 RASP

The Reconfigurable Analog Signal Processor, or RASP, is a platform for evaluating the design and implementation of floating-gate inspired FPAAs. Computational primitives with varying levels of granularity are used in order to completely cover the design space. A focus is on providing maximum reconfigurability. The first RASP was comprised of two CABs and a full cross-bar switch [50]. The second implementation of the RASP was also

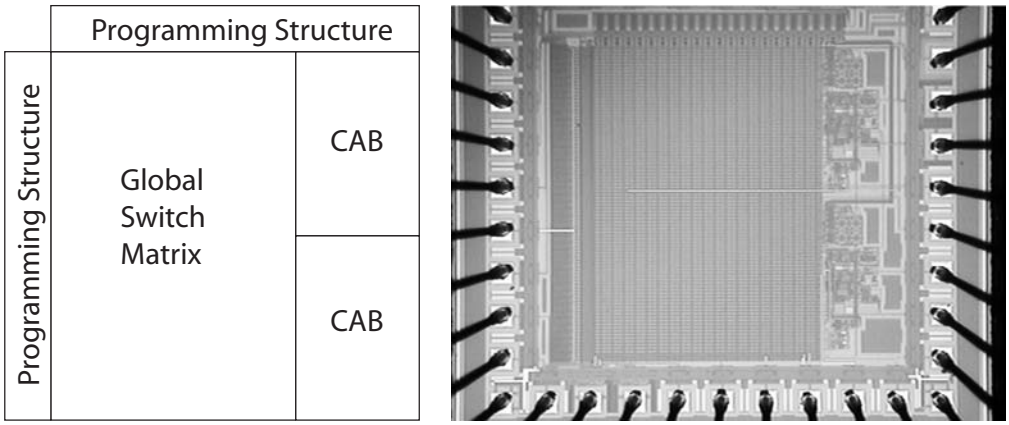


Figure 60. This is the architecture for the RASP 1.5. There are two CABs, a global cross-bar switch matrix, and programming circuitry. The contents of the CAB are displayed in Figure 61.

comprised of two CABs and a full cross-bar switch, but had architectural improvements necessary to produce meaningful circuit and system data. In addition to the description that follows, the implementation and results of that chip have been published as [51].

Referred to as the RASP 1.5, it is the system illustrated in Figure 60. Each CAB has three amplifiers, three filter caps, a min and max detector, a bandpass, a pFET, an nFET, and a vector matrix multiplier, as shown in Figure 61. The OTA is a 9-transistor wide-range amplifier with a floating-gate bias current. The max and min detectors have floating-gate elements for varying the time-constant of the max and min detection decay. The bandpass is a cascade of two compact capacitively coupled current conveyors, or C^4's, with a buffer in between to reduce loading.

7.1.1 VMM

The VMM and similar processing circuits, such as diffusors, are capable of utilizing large sections of the switch fabric for computation. As depicted by the 2x2 differential VMM structure of Figure 62, the VMM is composed almost entirely of programmed switch elements. The only CAB component is the OTA used to buffer the source voltage of the input switch element. The individual multiplier weights are set by programming a charge

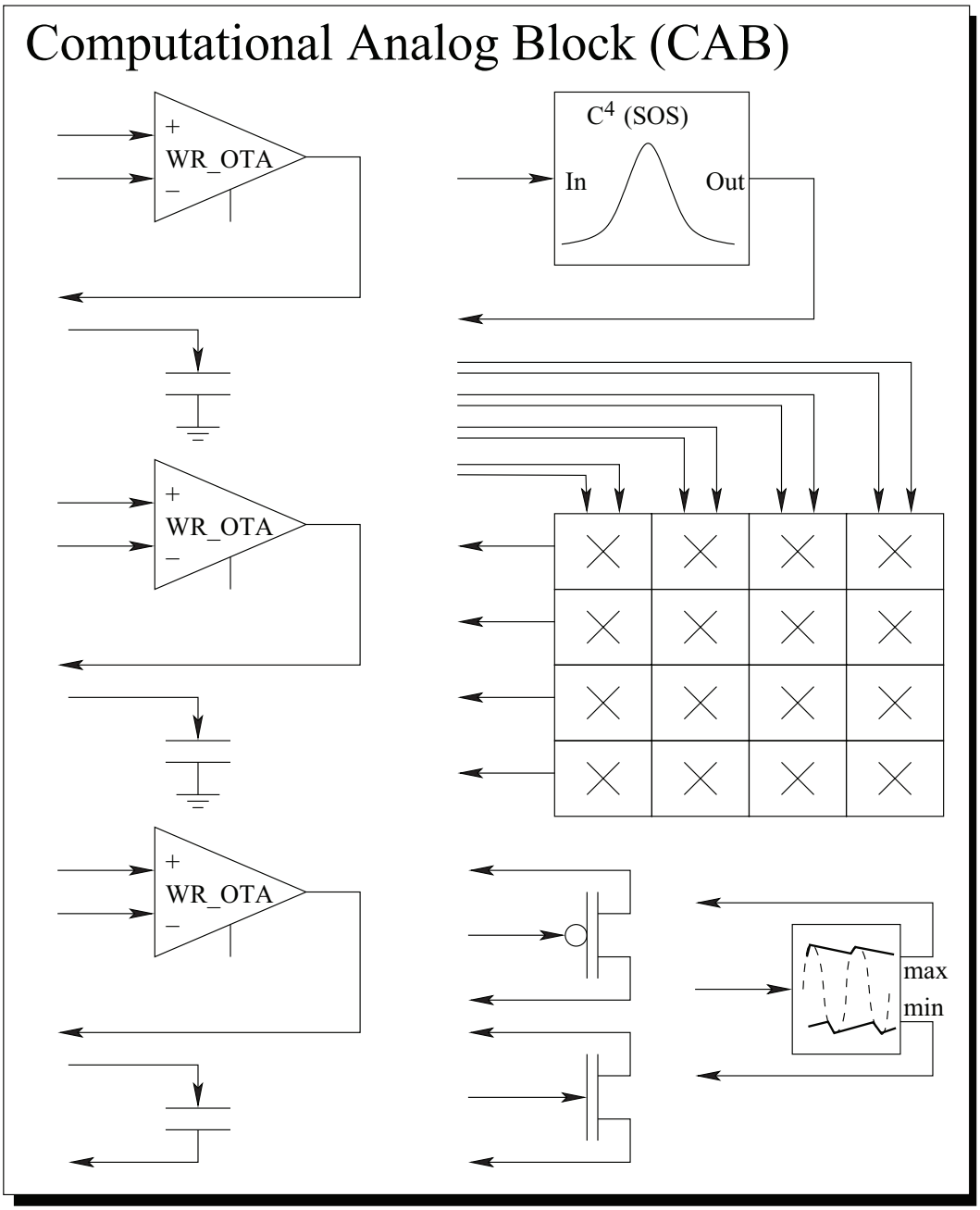


Figure 61. The components of a CAB on the RASP 1.5. The arrows represent connections to the switch matrix. The OTAs have a 9-transistor wide-range floating-gate biased architecture. The $C^4(SOS)$ block is a cascade of two C^4 circuits with a buffer in between. The max and min detectors have a τ set by a floating-gate.

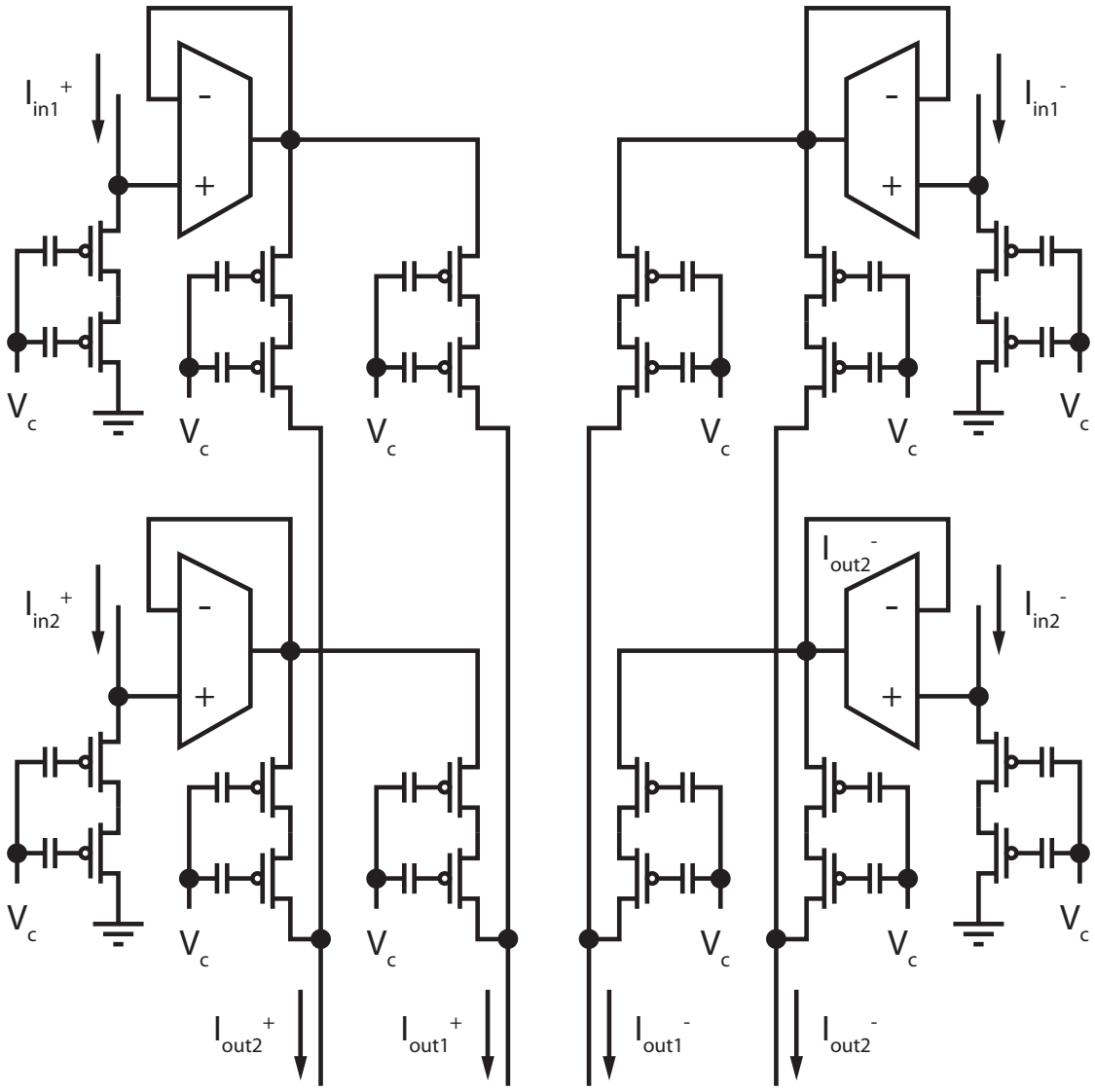


Figure 62. Differential 2x2 VMM structure utilizing programmable switch fabric elements (two floating gate switches in series).

difference between the input and output switch elements. Multiplier outputs are then tied together for current summation to perform the final computation. A two quadrant multiplier is constructed using a second single-ended multiplier to provide positive and negative inputs. In a similar fashion, a four quadrant multiplier can be constructed by duplicating the output stage of the two quadrant multiplier to provide positive and negative weights.

For a four quadrant structure, the CAB component utilization is dependent upon the number of inputs (#OTAs = 2*#inputs). The switch fabric utilization is dependent upon

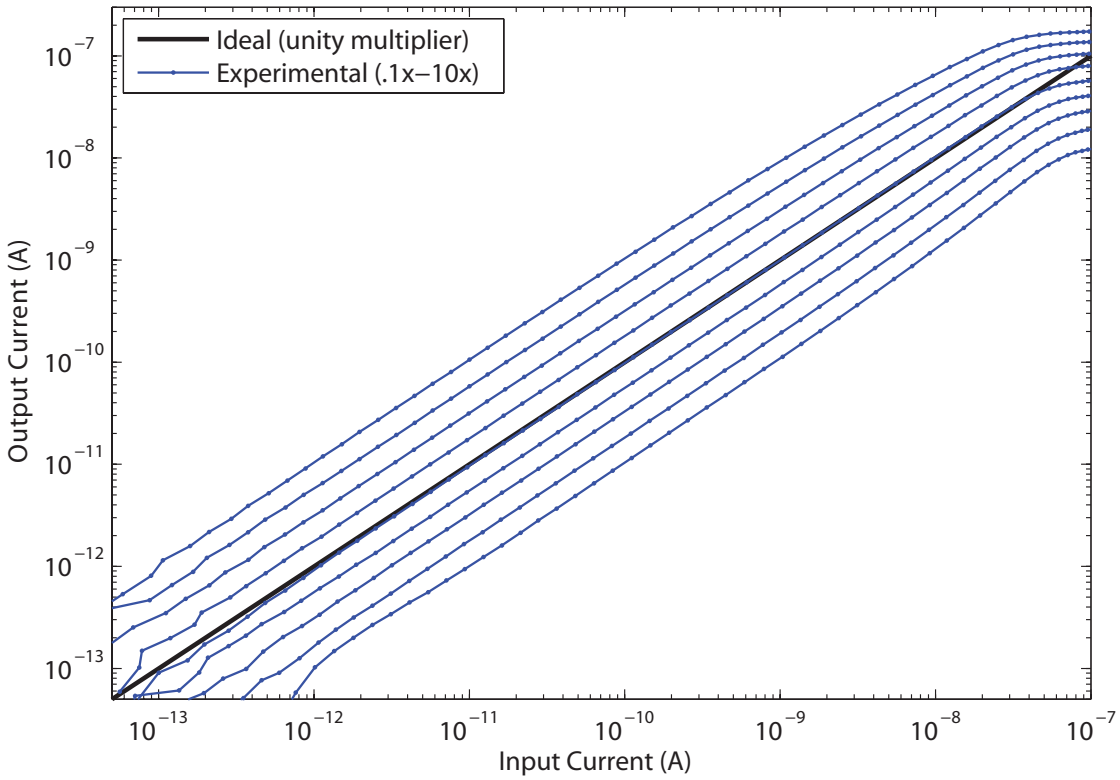


Figure 63. Single quadrant multiplication with weights programmed between .1 and 10.

both the inputs and outputs (#switch elements = 2*#inputs*[2*#outputs + 1]). Since there is no CAB component cost when increasing the number of outputs, additional processing can be added with little impact on CAB component utilization. Using this technique, arbitrarily sized VMMs can be constructed. This eliminates the need to include pre-sized versions of them as CAB components, which saves a significant amount of computational area for other components.

Data from a single multiplier is shown in Figure 63. This multiplier was constructed with a single floating gate switch element as an input, an amplifier buffering the source voltage, and another switch element as an output. The weight of the multiplier was programmed over two orders of magnitude from .1 to 10. The curvature apparent at higher currents is a result of the transistors leaving the subthreshold region of operation. The jagged profile at sub-picoamp currents results from limitations in the off-chip measurement

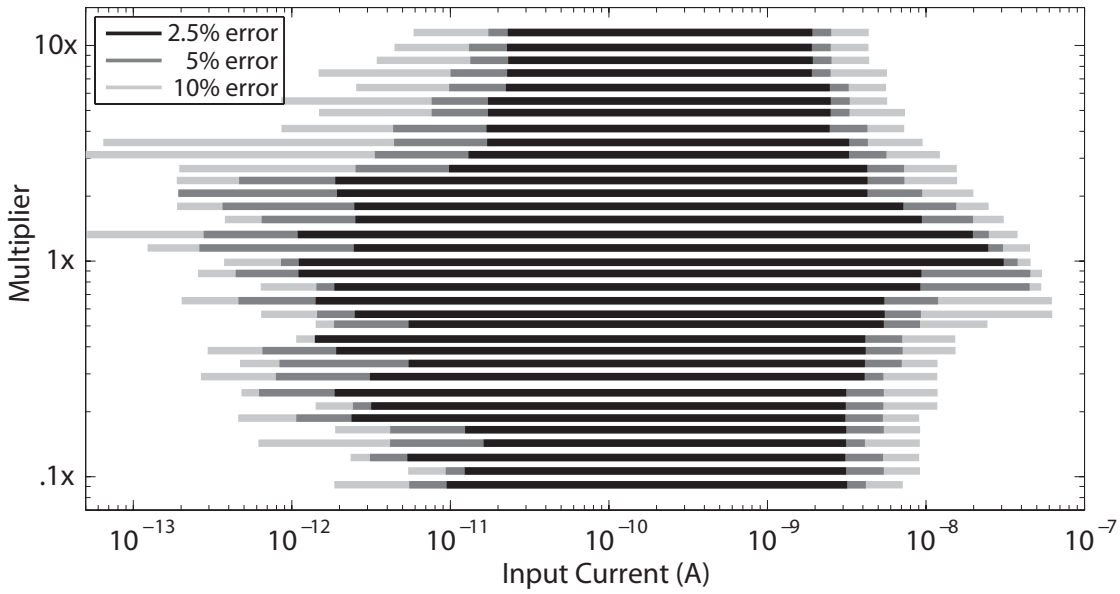


Figure 64. Plot depicting a range of multiplier values that produce outputs within a ± 2.5%, 5%, and 10% error band.

equipment. For multiplier weights ranging from .5 to 1.5, a reasonable range for common signal processing tasks, the error was observed to be within ±2.5% over three decades of current. The data of Figure 63 is analyzed in Figure 64 to illustrate the trade-off between accuracy and dynamic range in the multiplier element. The three error bands represent the range of currents over which a particular programmed multiplier results in an output that falls within the specified error range. As illustrated in Figure 64, the input range is greatest for all error bands around a unity multiplier. Since the circuit is functioning as a current mirror in this range, the effect of offsets between the input and output transistors is minimized.

7.1.2 Continuous-Time Filters

7.1.2.1 Follower-integrator

The implementation of follower-integrator is a good example of how to build circuits on a RASP FPAA. It is first necessary to tunnel the entire array in order to disconnect all of the switches and reset the bias positions in the array. Next, an operational transconductance amplifier (OTA) and capacitor is mapped to the switches in the connection matrix. The

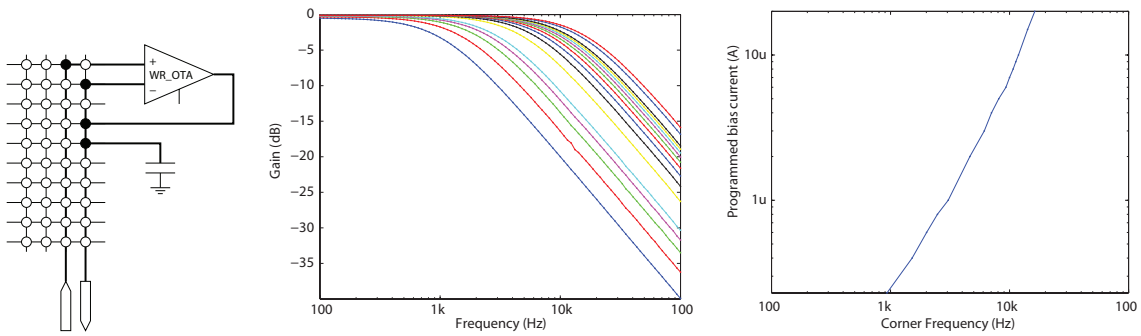


Figure 65. (a) Implementation of a follower-integrator in a cross-bar switch matrix. The black dots represent switches that have been injected into the *on* position. (b) Frequency response for several programmed currents. (c) Mapping of programmed current levels to corner frequencies.

resulting mapping is shown in Figure 65(a). The programming algorithm used for switches, Section 2.3.2.1, is then used to make the necessary circuit connections. Finally, the bias current of the amplifier is set by programming the floating-gate transistor to achieve a particular corner frequency.

Figure 65(b) is an illustration of the frequency responses taken from the follower-integrator circuit for several different programmed bias currents. In order to calibrate the $3db$ frequency against the parasitic capacitance in the switch matrix, the corner frequencies are extracted and mapped to the programmed bias currents. Shown in Figure 65(c), the mapping provides a way to quickly target a corner frequency. However, unless the output of the OTA-C circuit is buffered, the mapping will fail as other circuits are connected to its output.

7.1.2.2 Second-order Section

A slightly more complicated example of a circuit on the RASP is the second-order section, or SOS, pictured in Figure 66. This time all three OTAs and two caps from a single CAB are used to build the circuit. The circuit is desirable because it provides a low-pass transfer characteristic with a straightforward, predictable way to set the τ and Q factor. A small signal analysis of the circuit provides the means for relating the bias currents of the OTAs

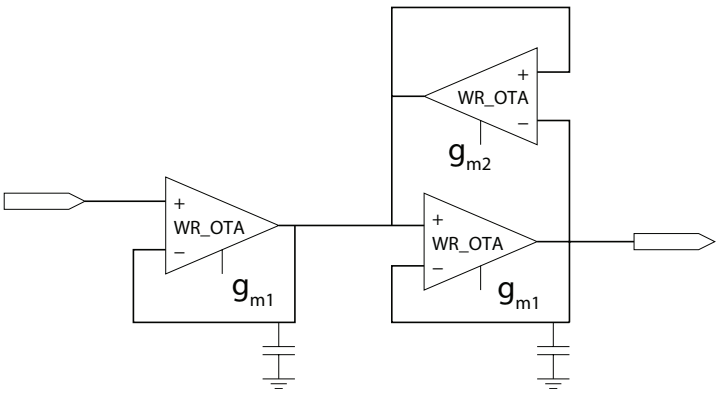


Figure 66. A second–order section filter can be implemented with two OTAs in a source-follower configuration and a third OTA that creates positive feedback.

to the transient response. The circuit has the transfer function

$$H(s) = \frac{1}{\tau^2 s^2 + \frac{1}{Q}\tau s + 1}\ ;\ \tau = \frac{C}{g_{m1}}\ ;\ Q = \frac{1}{2 - \frac{g_{m2}}{g_{m1}}} \tag{57}$$

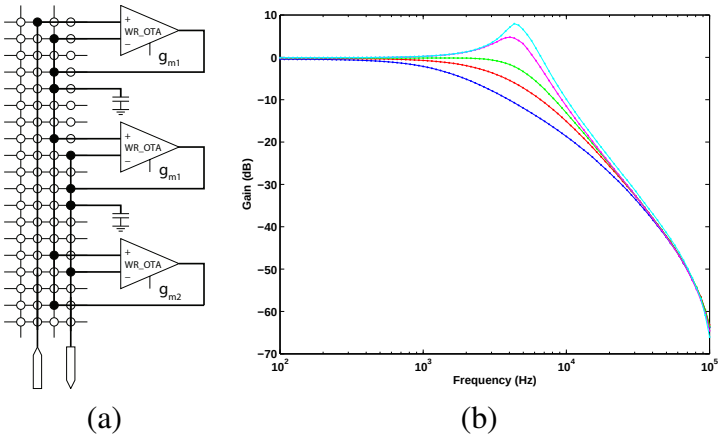
The τ of the SOS is the same as the follower-integrator, so the mapping from Figure 65c is still valid as long as the same capacitance is maintained. If not, the τ would have to be determined experimentally. In order to set the Q factor, a ratio of transconductances is necessary. In the case of two well matched, equivalently designed amplifiers, the ratio of transconductances is the ratio of the square-root of the bias currents. In subthreshold, it is simply the ratio of the bias currents.

The FPAA implementation and resulting data are shown in Figure 67. Data for a fixed g_{m1} and five different values of g_{m2} is shown. As expected, the Q factor increases with an increasing g_{m2}.

7.1.2.3 Ladder filter

The availability of OTAs and grounded capacitors makes the RASP ideal for implementing Gm-C filters, as demonstrated in the previous section. One way to realize a particular filter is by modeling it with resistors, inductors, and capacitors, and then synthesizing the design using Gm-C filters. In this example, a third-order Butterworth filter is implemented.

The canonical prototype of the filter, a double–resistance terminated LC filter, is shown



(a) (b)

Figure 67. SOS implementation and results. (a) The second-order section is implemented using the switch matrix, three OTAs, and two explicit capacitors. (b) The experimental frequency response of a circuit is shown here. Data for a fixed g_{m1} and five different values of g_{m2} is shown. As expected, the Q factor increases with an increasing g_{m2}.

in Figure 68a. By using the signal simulation method outlined in [52], the Gm-C filter shown in Figure 68b is generated. In order to maintain a maximally flat response, the following must hold: $2 * g_{m1} = g_{m2}$. Accordingly, the bias current of OTA-3 was set to half of the other OTA bias currents. A range of bias currents was used to create the frequency response shown in Figure 68c. As expected, the corner frequency of the filter is proportional to the bias currents of the OTAs. The lower corners were obtained by using a bias current in the range of hundreds of pico-amps, while the highest corners required currents of up to 1 μA.

7.2 RAAM

The Reconfigurable Analog Array of MITEs, or RAAM, is another test-bed FPAA built upon the same switch technology as the RASP. However unlike the RASP, the RAAM represents a concerted effort to better mimic FPGA design by using a regular computational primitive, explicitly supporting direct system synthesis, and having multiple levels of connection hierarchy.

MITEs were conceived and developed by Brad Minch. In [53], he presents a number of different ways to implement MITEs, including the floating gate implementation pictured

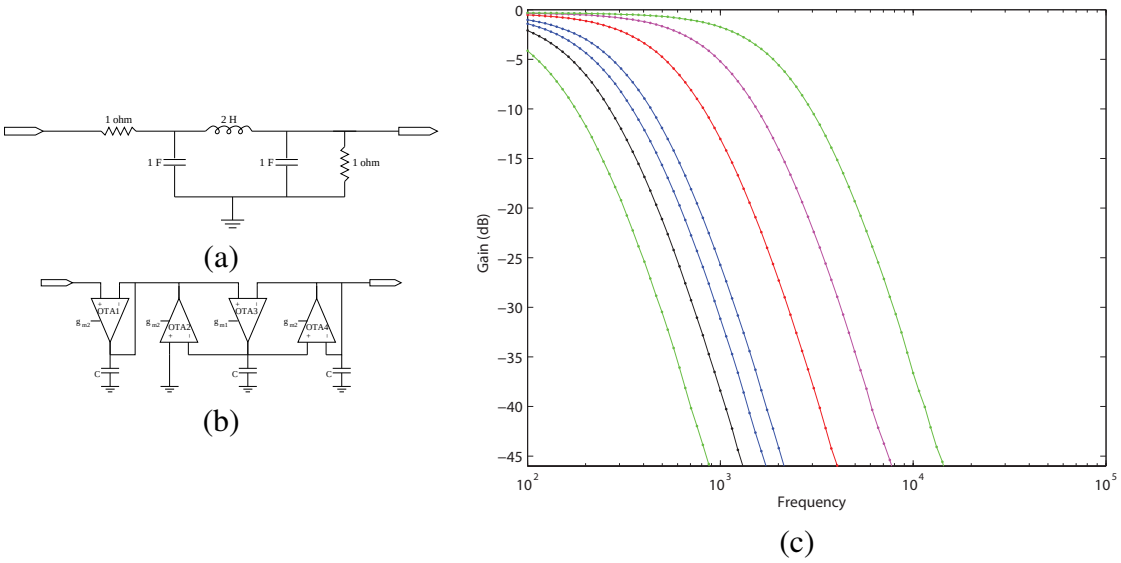


Figure 68. (a) The canonical prototype of a third–order Butterworth double–resistance terminated LC filter. (b) This is the Gm-C implementation of the same filter. The filter can be realized directly on the RASP 1.5 FPAA. (c) Results from the ladder filter for different bias currents.

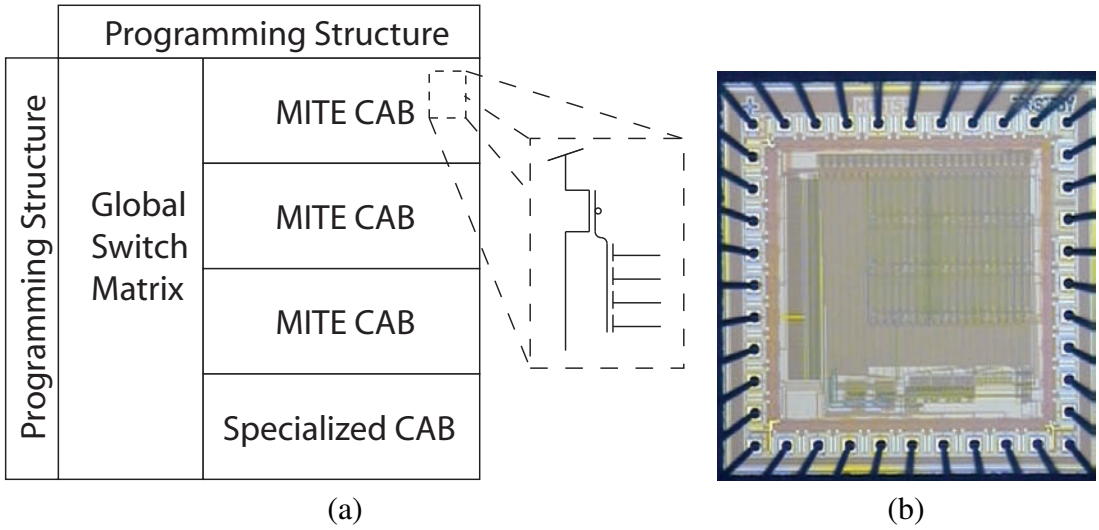


Figure 69. System architecture of the RAAM, an FPAA used to create reconfigurable translinear networks. The system consists of 3 MITE CABS, a specialized CAB, and a global switch network. The specialized cab consists of circuitry that enables dynamic functions and also includes an input bank of V-I converters.

in Figure 69a. An ideal MITE has K input voltages. Each voltage is scaled by w_K, a positive dimensionless weight. The exponential result of the weighted sum is scaled and represented as a current. An idealized expression for the MITE is provided in (58).

$$I = I_s e^{\sum w_i V_i} \tag{58}$$

MITEs map well to higher-level system descriptions by leveraging the power of translinear circuit methodology. With respect to general translinear field, there is a lot work covering the synthesis of static and dynamic translinear circuits. At least two synthesis procedures have been developed specifically for MITEs [54, 55]. In [55], the synthesis procedure allows mapping from single output static polynomial constraints and algebraic differential equations to MITE circuits. Static and dynamic systems are treated in a similar manner. [55] relates multiple inputs and multiple outputs of a static mathematic expression to a connectivity matrix which is then mapped to MITEs. The dynamics of the system are mapped to first-order low-pass filters. It implements a smaller class of dynamic circuits, but provides a more direct mapping to an FPAA fabric.

The RAAM has four CABs, a global switch matrix, and programming circuitry. It is pictured in Figure 69 and was fabricated in a $.5\mu$ process. The global switch matrix is actually the second layer of hierarchy–the MITE CAB is composed of MITE primitives and a local switch matrix. The purpose of the local network is provide a trade-off between switch area and reconfigurability. There are two analog primitives in the MITE CAB, a diode connected MITE for input signals and a plain MITE for output signals. The 4-cap structure was chosen as a means for mapping cleanly to the synthesis procedure in [55]. The specialized CAB contains the I-V converters for inputs and first-order low-pass filters for implementing dynamic circuits.

The following experimental results illustrate the basic functionality of the RAAM architecture.

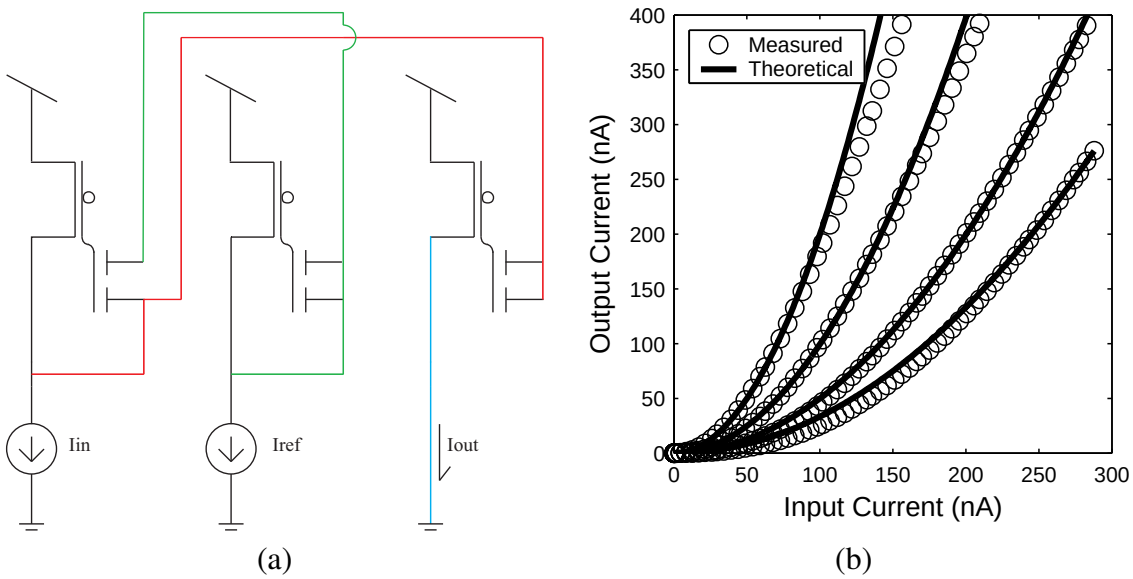


Figure 70. Schematic of a squaring circuit represented by (59) **using two input MITEs and a single output MITE. The coloration corresponds to connections made in order to implement the circuit on the RAAM. The mapping is illustrated in Figure 71.**

7.2.1 Single-input power-law circuit

A MITE is fundamentally well-suited for implementing power-law equations. Accordingly, a good starting point with the RAAM is the implementation of a circuit that results in an input raised to a particular power. Using [55], $y = x^2$ can be represented as

$$I_{out} = \frac{I_{in}^2}{I_{ref}} \tag{59}$$

where I_{out} is the output current, I_{in} is an input current, and I_{ref} is a scaling current which represents unity. The circuit that represents (59) is shown in Figure 70a. By defining V_{ref} as the diode-connected voltage created by I_{ref}, V_{in} as the diode-connected voltage created by I_{in}, and $w = -\frac{\kappa}{U_T}\left(\frac{C}{C_T} + \frac{Q}{C_T}\right)$, the following expressions for the controlling voltages can be written from the circuit in Figure 70:

$$V_{ref} = \frac{1}{2w}\ln\left(\frac{I_{ref}}{I'_s}\right) \; ; \; V_{in} = \frac{1}{w}\ln\left(\frac{I_{in}}{I'_s}\right) - V_{ref} \tag{60}$$

As a result,

$$I_{out} = I'_s e^{2wV_{in}} = I'_s e^{2\ln\left(\frac{I_{in}}{I'_s}\right) - \ln\left(\frac{I_{ref}}{I'_s}\right)} = I'_s e^{2\ln\left(\frac{I_{in}}{I'_s}\right) - \ln\left(\frac{I_{ref}}{I'_s}\right)} \tag{61}$$

which can be simplified to (59).

The analysis yields important insight. In order to cleanly simplify the result, the weight term must be equal for all three MITEs. If there is any variation in the weight term, it appears in the final expression as an additional exponent in the same way the squared term does. Accordingly, it is necessary to equalize the charge on the floating gates, maintain good capacitor matching, and avoid large temperature gradients across the circuit operation. It is also possible to offset some of the mismatch by adjusting the charge on the output MITE, as is shown in Section 7.2.2.

The square circuit was compiled into the RAAM yielding the experimental data plotted against simulation data in Figure 70b. Reference currents of $50nA$, $100nA$, $200nA$, and $300nA$ were used. The circuit is implemented by mapping the 2-cap MITE circuit to 4-cap MITEs in a RAAM CAB. The resulting implementation is shown in Figure 71. Two currents are routed to two input MITEs and an output MITE. The colored circles at line intersections represent switches that have been injected to the on position. The output MITE uses a cascoded nFET current mirror in order to reduce distortion in the current mirror. Variations due to differences in the current mirror can be taken care of by varying the charge on the output MITE.

7.2.2 Vector magnitude

As an example of a slightly more complicated system, a MITE circuit that calculates the vector magnitude was compiled onto the RAAM. The equation for the circuit is given by

$$I_{out} = \sqrt{I_x^2 + I_y^2} \tag{62}$$

The inputs provided to the system in the form

$$I_x = I_{ref} * cos(\theta) \tag{63}$$

$$I_y = I_{ref} * sin(\theta) \tag{64}$$

where θ is swept from 0 to 90°. A plot of the initial system implementation is shown in Figure 72a. Each concentric arch represents a different I_{ref}. Empirically, the data has a

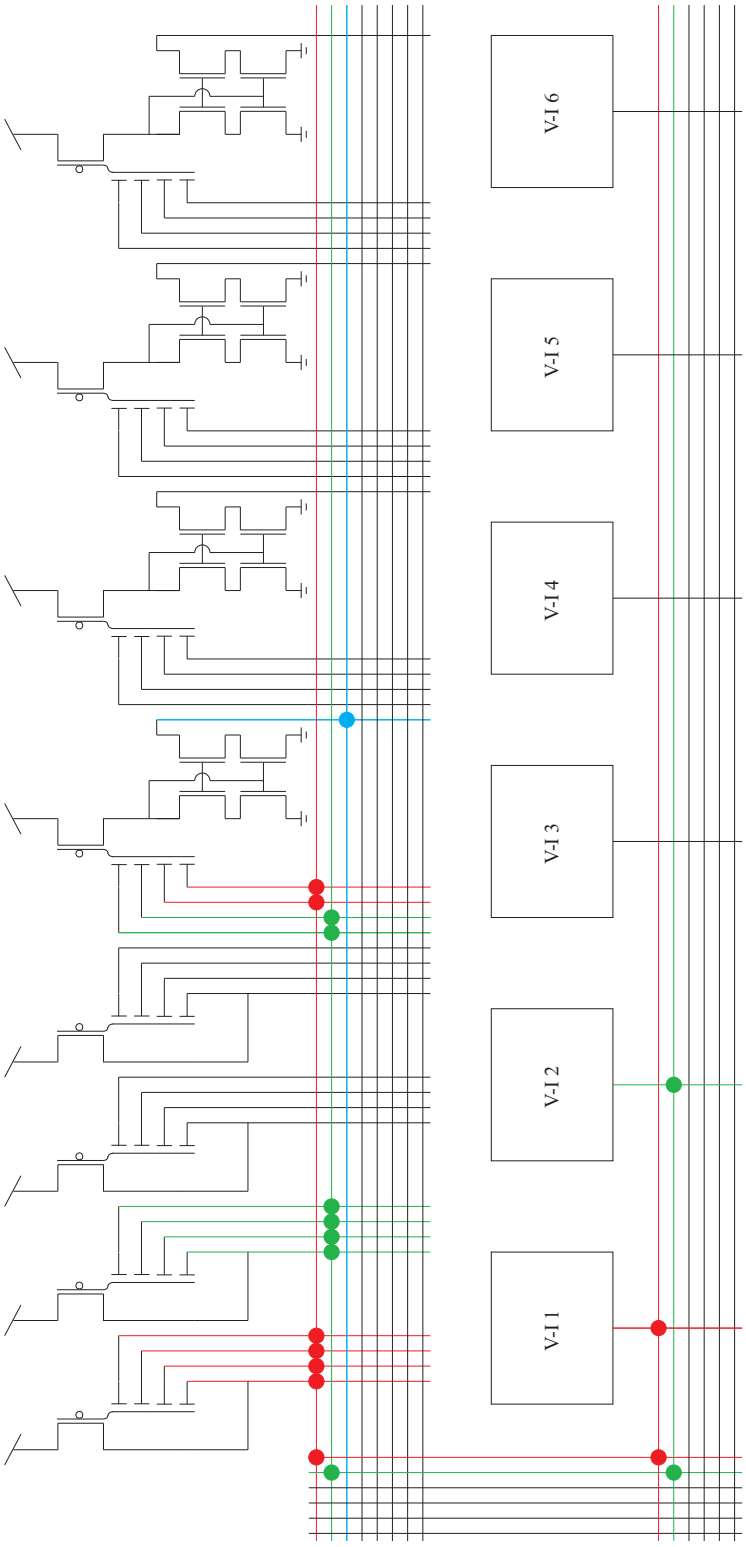


Figure 71. Example of the RAAM reconfigured to implement a squaring circuit. The colored nodes correspond to Figure (70) **and the circles at the intersection of the bus lines indicate a switch that has been turned on. The row of V-I converters and the crossbar network below it represent the specialized CAB, the crossbar network on the left of the figure represents the global switch matrix, and the row of MITEs and the crossbar network below it represent a MITE CAB.**

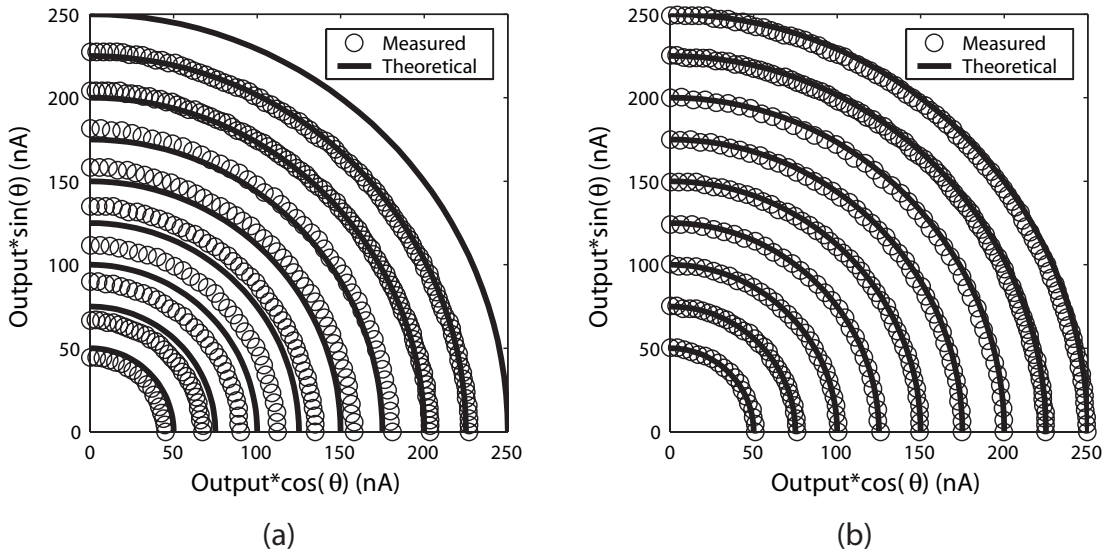


Figure 72. Results of the vector magnitude circuit. (a) Results of the vector magnitude circuit after programming all MITEs to the same level. Each MITE was programmed to have $10nA$ of current with a source-drain voltage of $2.3V$ and a source-gate voltage of $1.3V$. (b) Results of the vector magnitude circuit after programming out the initial errors. The MITEs preforming the squaring functions were injected higher than the other MITEs in order to increase the coefficients to 1.

coefficient under the square-root that is modeled as

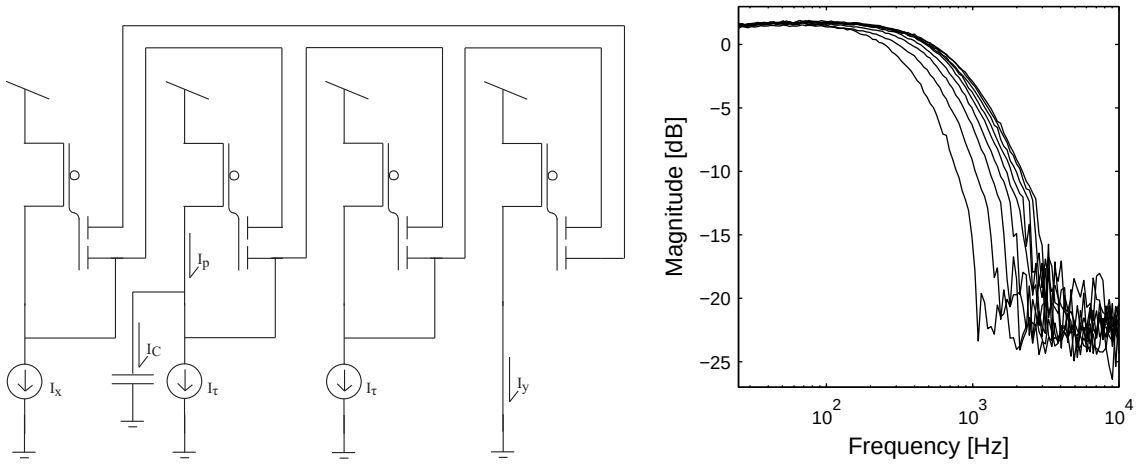
$$I_{out} = \sqrt{0.8I_x^2 + 0.8I_y^2}. \tag{65}$$

By increasing the charge of the MITEs implementing the square-root, the empirical coefficient is increased to unity. The resulting data is shown in Figure 72b.

7.2.3 First-order filter

MITEs can be used to implement more than static equations. By adding a 1^{st}-order low-pass filter to a static MITE network, algebraic differential equations can be implemented [54, 55]. If I_x is an input current and I_y is an output current equation, a low-pass filter can be expressed as

$$\tau \frac{dI_y}{dt} + I_y = I_x \tag{66}$$



(a) (b)

Figure 73. MITE implementation, (a), and experimental data, (b), of a 1^{st}-order low-pass filter. The filter is simply a translinear loop with a capacitor on an internal node.

However, capacitors are related to current through the time-derivative of a voltage, so the chain rule is used to rearrange (66) into

$$\tau \frac{\delta I_y}{\delta V_y} \frac{dV_y}{dt} + I_y = I_x \tag{67}$$

In addition, the voltage-derivative of I_y is simply the g_m of a MITE, resulting in

$$-w \frac{\tau I_y}{U_T} \frac{dV_y}{dt} + I_y = I_x \tag{68}$$

After introducing $\frac{C}{C}$ to the time-derivative of V_y and rearranging,

$$I_\tau - I_C = \frac{I_x I_\tau}{I_y} = I_p \tag{69}$$

Accordingly, (69) can be directly implemented as a translinear loop with a capacitor on one of the inputs.

The resulting circuit implementation is shown in Figure 73a. The filter was implemented on the RAAM with the resulting experimental data shown in Figure 73b.

CHAPTER 8

CONCLUSION

Over the course of this document, I have shown how to understand and interact with floating-gate transistors in a way that informs and enables building large-scale analog reconfigurable systems. I explained the key charge movement mechanisms and how those physical processes work both for and against us in the context of dense arrays of floating-gate transistors. I used the understanding of charge injection in a floating-gate pFET to build a trend-accurate low parameter count simulation model for injection simulation. I further showed how to combine an understanding of charge injection and my simulaton methodology to design and implement a compact computational cell. I then combined the different floating-gate techniques into two different targeted reprogrammable analog systems, a computational analog transform image sensor and an analog adaptive filter. I subsequently constructed two different types of fully reconfigurable analog systems, a reconfigurable analog signal processor based on a variety of analog components with different granularity, and an analog computational system with a specific algorithmic targeting in mind.

8.1 Specific Contributions

I designed, implemented and tested a two CAB general purpose FPAA along with Chris Twigg and Tyson Hall. I completed design and layout of some of the CAB components, and worked with Chris on the programming logic. The chip laid the groundwork for all of the subsequent FPAAs used in our research group and was published in [51]. Based on the efforts with the generic FPAA, I designed, implemented, and tested a two CAB MITE FPAA with Dave Abramson and Shyam Subramanian. I used my work on the general purpose FPAA to build a MITE FPAA with assistance from Dave. We subsequently constructed a second MITE chip using lateral BJTs. Shyam provided the algorithmic approach

for mapping the equations we synthesized in hardware and our work was published in [56]. We also looked at applying the MITES to particle filtering in collaboration with Rajbabu Velmurugan, published in [57]. I investigated the programming and characterization of a floating-gate switch with Chris and Dave. My approach to floating-gate switch programming enabled the results shown in the previous two papers, and my characterization work comparing floating-gate switches to pass gates and T-gates is published in [58]. In order to address the questions about the "leakiness" of floating-gate analog memory, I looked at long-term charge storage in the context of floating-gate offset removal with Venkatesh Srinivasan and Guillermo Serrano. I helped research the charge leakage mechanism, construct the test platform, and analyze the data. Our results are published in [38]. In parallel, I used the insight from my work with floating-gate switch programming and isolation to do a detailed study of parasitic charge movement, where I identified that subthreshold conduction was not the dominant parasitic charge mechanism in our floating-gate arrays, which I published in [59]. As the research progressed, I found myself limited by a lack of a floating-gate simulation model, so I built a simplified implementation of CHE injection in Verilog-A based based on a physical (rather than empirical) model, and fit it to experimental data. That effort was published in as [60]. I then turned my attention to analog current-mode multiplication, first in the context of an analog adaptive filter with Venkatesh Srinivasan. Our work was published in [61]. I next looked at the multiplier in the switch fabric of an FPAA with a vector-matrix multiplier in mind. Chris and I investigated how to better utilize our FPAA architecture for computation in [62], where I applied the in-switch VMM effort. The in-switch VMM approach has a profound impact on the utilization of chip area for computation. During the VMM investigation for the FPAA, I designed, implemented, and tested a matrix-vector multiplier using floating-gate transistors in a computational image sensor. I helped design the log-amp buffering the input signal and the log-amp buffering the output signals, along with Dave and Ryan Robucci, and I designed the array and programming for the VMM. I also contributed to the programming on the front-end of the image sensor.

That effort as been published in [63].

REFERENCES

[1] D. B. SCHWARTZ, R. E. HOWARD, and W. E. HUBBARD, "A programmable analog neural network chip," *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, vol. 24, no. 2, pp. 313–319, 1989.

[2] F. J. KUB, K. K. MOON, I. A. MACK, and F. M. LONG, "Programmable analog vector matrix multipliers," *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, vol. 25, no. 1, pp. 207–214, 1990.

[3] T. SHIBATA and T. OHMI, "A functional mos-transistor featuring gate-level weighted sum and threshold operations," *IEEE TRANSACTIONS ON ELECTRON DEVICES*, vol. 39, no. 6, pp. 1444–1455, 1992.

[4] J. RAMIREZANGULO, S. C. CHOI, and G. GONZALEZALTAMIRANO, "Low-voltage circuits building-blocks using multiple-input floating-gate transistors," pp. 971–974, 1995.

[5] B. A. Minch, C. Diorio, P. Hasler, and C. A. Mead, "Translinear circuits using subthreshold floating-gate mos transistors," *ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING*, vol. 9, no. 2, pp. 167–179, 1996.

[6] P. Hasler and T. S. Lande, "Overview of floating-gate devices, circuits, and systems," pp. 1–3, 2001.

[7] J. Ramirez-Angulo and A. J. Lopez, "Mite circuits: The continuous-time counterpart to switched-capacitor circuits," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II-ANALOG AND DIGITAL SIGNAL PROCESSING*, vol. 48, no. 1, pp. 45–55, 2001.

[8] R. Benson and D. Kerns, "Uv-activated conductances allow for multiple time scale learning," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 434 – 40, May 1993.

[9] E. Takeda, A. Shimizu, and T. Hagiwara, "Role of hot-hole injection in hot-carrier effects and the small degraded channel region in mosfet's," *IEEE Electron Device Letters*, vol. 4, no. 9, pp. 329–331, 1983.

[10] J.-H. Chen, S.-C. Wong, and Y.-H. Wang, "An analytic three-terminal band-to-band tunneling model on gidl in mosfet," *IEEE Transactions on Electron Devices*, vol. 48, no. 7, pp. 1400 – 5, July 2001.

[11] Y. Taur and T. H. Ning, *Fundamentals of modern VLSI devices*. New York, NY, USA: Cambridge University Press, 1998.

[12] P. Hasler, B. Minch, and C. Diorio, "Adaptive circuits using pfet floating-gate devices," *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*, pp. 215 – 29, 1999.

[13] G. Serrano, P. Smith, H. Lo, R. Chawla, T. Hall, C. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating-gate elements," *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. Vol.1, pp. 373 – 6, 2004.

[14] V. Srinivasan, G. Serrano, J. Gray, and P. Hasler, "Precision cmos amplifier using floating-gate offset cancellation," *IEEE Custom Integrated Circuits Conference*, pp. 739–743, 2005.

[15] H. Nozama and S. Kokyama, "A thermionic electron emission model for charge retention in SAMOS structures," *Japanese Journal of Applied Physics*, vol. 21, pp. L111–L112, Feburary 1992.

[16] C. Bleiker and H. Melchior, "A four-state EEPROM using floating-gate memory cell," *IEEE Journal of Solid-State Circuits*, vol. 22, pp. 460–463, Jun. 1987.

[17] S. Chakrabartty and G. Cauwenberghs, "Sub-microwatt analog vlsi trainable pattern classifier," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 5, pp. 1169 – 79, 5 2007.

[18] R. Robucci, L. Chiu, J. Gray, J. Romberg, P. Hasler, and D. Anderson, "Compressive sensing on a cmos seperable transform image sensor," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2008.

[19] C. Twigg and P. Hasler, "A large-scale reconfigurable analog signal processor (rasp) ic," *IEEE Custom Integrated Circuits Conference*, September 2006.

[20] J. Brown, W. D. Brewer, *A Comprehensive Guide to Understanding and Using NVSM Devices*. Wiley-IEEE Press, 1997.

[21] P. D. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *Proceedings of the International Symposium on Circuits and Systems*, 2002, pp. 489–492.

[22] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, pp. 305 – 327, 2003.

[23] P. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *International Symposium on Circuits and Systems*, vol. 5, Phoenix, AZ, May 2002, pp. 489–492.

[24] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Adaptive algorithm using hot-electron injection for programming analog computational memory elements within 0.2 percent of accuracy over 3.5 decades," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 2107 – 2114, September 2006.

[25] C. Twigg and P. Hasler, "Programmable conductance switches for fpaas," *IEEE International Symposium on Circuits and Systems*, pp. p173–176, September 2007.

[26] S. Chakrabartty and G. Cauwenberghs, "Fixed-current method for programming large floating-gate arrays," *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 3934 – 3937, May 2005.

[27] P. Hasler, A. Basu, and S. Koziol, "Above threshold pfet injection modeling intended for programming floating-gate systems," *IEEE International Symposium on Circuits and Systems*, pp. 4 pp. –, 2007.

[28] K. Rahimi, C. Diorio, C. Hernandez, and M. Brockhausen, "A simulation model for floating-gate mos synapse transistors," *IEEE International Symposium on Circuits and Systems. Proceedings*, vol. vol.2, pp. 532 – 5, 2002.

[29] R. Sarpeshkar, "Efficient precise computation with noisy components: Extrapolating from an electronic cochlea to the brain," Ph.D. dissertation, California Institute of Technology, 1997.

[30] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nw/mhz, 128×32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004*, 3-6 Oct. 2004, pp. 651–654.

[31] B. Minch, "Floating-gate techniques for assessing mismatch," in *IEEE International Symposium on Circuits and Systems*, 2000.

[32] T. Serrano-Gotarredona, B. Linares-Barranco, and A. Andreou, "Very wide range tunable cmos/bipolar current mirrors with voltageclamped input," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 46, pp. 1398–1407, 1999.

[33] S. Chakrabartty, G. Singh, and G. Cauwenberghs, "Hybrid support vector machine/hidden markov model approach forcontinuous speech recognition," in *IEEE Midwest Symposium on Circuits and Systems*, 2000.

[34] R. Robucci, "Development of a computational image sensor with applications in integrated sensing and processing," Ph.D. dissertation, Georgia Institute of Technology, 2009.

[35] C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead, "A high-resolution nonvolatile analog memory cell," in *IEEE International Symposium on Circuits and Systems*, 1995.

[36] E. Ozalevli, C. Twigg, and P. Hasler, "10-bit programmable voltage-output digital-analog converter," in *International Symposium on Circuits and Systems*, 2005.

[37] G. Serrano and P. Hasler, "A precision low tc wide range cmos current reference," *IEEE Journal of Solid-State Circuits*, 2008.

[38] V. Srinivasan, G. Serrano, J. Gray, and P. Hasler, "A precision cmos amplifier using floating-gate transistors for offset cancellation," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 2, pp. 280 – 91, Feb 2007.

[39] R. McFadyen and F. Schlereth, "Gain-compensated logarithmic amplifier," in *Solid-State Circuits Conference. Digest of Technical Papers. 1965 IEEE International*, vol. VIII, Feb 1965, pp. 110–111.

[40] A. Basu, R. W. Robucci, and P. E. Hasler, "A low-power, compact, adaptive logarithmic transimpedance amplifier operating over seven decades of current," *Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]*, vol. 54, no. 10, pp. 2167–2177, 2007.

[41] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 834–845, May 2005.

[42] ——, "Correlation learning rule in floating-gate pFET synapses," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 65–73, Jan. 2001.

[43] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531nw/mhz, 128x32 current-mode vector matrix multiplier with over 2 decades of linear range," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 29–4–1 – 29–4–4, Oct. 2004.

[44] Y. Tsividis and S. Satyanarayana, "Analogue circuits for variable synapse electronic neural networks," *Electronics Letters*, vol. 24, no. 2, pp. 1313–1314, 1987.

[45] I. A. Mack, F. Kub, K. K. Moon, and F. M. Long, "Programmable Analog Vector-Matrix Multiplier," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 207–214, Feb. 1990.

[46] A. J. Agranat, C. F. Neugebauer, R. D. Nelson, and A. Yariv, "The CCD neural processor: A neural network integrated circuit with 65536 programmable analog synapses," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 8, pp. 1073–1075, Aug. 1990.

[47] P. Hafliger and C. Rasche, "Floating-gate analog memory for parameter and variable storage in a learning silicon neuron," *Proceedings of the International Symposium on Circuits and Systems*, pp. 416–419, May 1999.

[48] G. Gomez and R. Siferd, "Single-chip FIR adaptive filter using CMOS analog circuits," *Proceedings of the IEEE International ASIC Conference and Exhibit*, pp. P3–5.1–P3.5.4, Sep. 1991.

[49] V. Srinivasan, R. Chawla, and P. Hasler, "Linear current-voltage and voltage-current converters," *Proceedings of the Midwest Symposium on Circuits and Systems*, pp. 675–678, Aug. 2005.

[50] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Application performance of elements in a floating–gate FPAA," in *Proceedings of the International Symposium on Circuits and Systems*, 2004, pp. 589–592.

[51] T. Hall, C. Twigg, J. Gray, P. Hasler, and D. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 52, no. 11, pp. 2298 – 307, Nov 2005.

[52] Y. Sun, *Design of high frequency integrated analogue filters*. The Institution of Engineering and Technology, London, UK, 2002.

[53] B. A. Minch, "Analysis, synthesis, and implementation of networks of multiple-input translinear elements," Ph.D. dissertation, California Institute of Technology, May 1997.

[54] ——, "Synthesis of static and dynamic multiple-input translinear element networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 2, pp. 409–421, Feb. 2004.

[55] S. Subramanian, D. Anderson, and P. Hasler, "Synthesis of static multiple input multiple output mite networks," *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, pp. I–189– I–192, 2004.

[56] D. Abramson, J. Gray, S. Subramanian, and P. Hasler, "A field-programmable analog array using translinear elements," in *System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop on*, 2005, pp. 425–428.

[57] R. Velmurugan, S. Subramanian, V. Cevher, D. N. Abramson, K. Odame, J. Gray, H. Lo, J. McClellan, and D. Anderson, "On low-power analog implementation of particle filters for target tracking," in *Signal Processing, 2006. EUSIPCO '06. Proceedings of the14th European Conference on*, 2006.

[58] J. Gray, C. Twigg, D. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pfet switches in an fpaa crossbar network," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 468–471 Vol. 1.

[59] J. Gray and P. Hasler, "Parasitic charge movement in floating-gate array programming," *Submitted to IEEE International MWSCAS*, 2008.

[60] J. Gray, R. Robucci, and P. Hasler, "The design and simulation model of an analog floating-gate computational element for use in large-scale analog reconfigurable systems," *Submitted to IEEE International MWSCAS*, 2008.

[61] J. Gray, V. Srinivasan, R. Robucci, and P. Hasler, "A floating-gate transistor based continuous-time analog adaptive filter," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008.

[62] C. Twigg, J. Gray, and P. Hasler, "Programmable floating gate fpaa switches are not dead weight," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 2007.

[63] R. Robucci, J. Gray, D. Abramson, and P. Hasler, "A 256 x 256 separable transform cmos imager," in *IEEE International Symposium on Circuits and Systems*, 2008.